UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de
Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE MONITORIZACIÓN DE SERVICIOS DE VÍDEO SOBRE REDES IP CON SEÑALIZACIÓN RTSP

Marta Montilla Carrascosa
Tutor: Jorge E. López de Vergara Méndez

JULIO 2018

DESARROLLO DE UN SISTEMA DE MONITORIZACIÓN DE SERVICIOS DE VÍDEO SOBRE REDES IP CON SEÑALIZACIÓN RTSP

AUTOR: Marta Montilla Carrascosa
TUTOR: Jorge E. López de Vergara Méndez

High Performance Computing and Networking Research Group
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2018

Resumen

En los últimos años, los sistemas de distribución de video sobre redes IP han ido adquiriendo gran relevancia, de tal manera que muchos operadores de red ofrecen en la actualidad este tipo de servicios. Por ello, cada vez es más necesario y urgente disponer de herramientas de monitorización en tiempo real de los mismos.

En este proyecto hemos desarrollado una herramienta de este tipo, que hace un seguimiento en tiempo real de los servicios de VoD (Vídeo bajo demanda) que utilizan una distribución *unicast* y recurren al protocolo de señalización RTSP para intercambiar mensajes entre el cliente y el servidor multimedia.

El programa desarrollado informa en tiempo real si ha habido alguna incidencia en la sesión y muestra la evolución de la misma. Además, identifica el protocolo de transporte y los puertos del servidor y del cliente que se utilizan en la transmisión del flujo multimedia. Finalmente, el programa se encarga de realizar las medidas de calidad oportunas sobre dicho flujo y emite un informe al respecto.

En concreto, el estudio de calidad se basa en la cuantificación de las pérdidas de paquetes en el flujo multimedia a través del indicador de continuidad de la cabecera MPEG-TS. Nuestro programa reporta los resultados sobre pérdidas de manera periódica, a intervalos de tiempo ajustables, lo que permite evaluar en tiempo real si está habiendo una mala calidad en la trasmisión. De este modo, si se produjera alguna incidencia se podría identificar y actuar inmediatamente para resolverla.

Por último, se ha validado el programa desarrollado realizando diferentes pruebas sobre capturas reales de tráfico tomadas de la red.

Palabras clave: RTSP, IPTV, MPEG-TS, Calidad de Servicio, pérdida de paquetes.

Abstract

In the last years, the systems for delivery of video contents over IP networks have become very relevant. Therefore, nowadays many network operators offer this kind of services. Thus, it is more and more necessary and urgent to have real time monitoring tools for them.

In this project, we have developed a tool for real-time checking of VoD (Video on Demand) services. These services utilize a unicast distribution and use the RTSP signaling protocol to exchange messages between the client and the multimedia server.

The developed program informs in real-time whether there has been any incidence during the session and shows its evolution. Additionally, it identifies in which transport protocol and ports the multimedia flow is transmitted, and it makes quality measurements on it, providing the corresponding report.

Specifically, the quality study is based in quantifying the loss of packets along the multimedia stream through the continuity counter, present in the MPEG-TS header. Our program reports periodically the loss results, in time intervals chosen at will. This allows the evaluation in real-time if there is a bad-quality transmission. This way, if there is any incidence, it could be identified and it is possible to act immediately to solve it.

Finally, we have checked our program using several real traffic traces taken from the network.

Keywords: RTSP, IPTV, MPEG-TS, Quality of Service, packet loss.

Agradecimientos

Quisiera expresar en estas líneas mi agradecimiento a todas las personas que han hecho posible y me han ayudado durante la realización de este Trabajo Fin de Grado.
En primer lugar, a mi tutor del TFG Jorge E. López de Vergara Méndez que aceptó dirigirme este trabajo y me ha acompañado a lo largo de todo el año. Le agradezco su apoyo, amabilidad, disponibilidad y todo lo que he aprendido con él.

Además el trabajo de fin de grado como culminación de los estudios es un buen momento para agradecer el apoyo y el cariño de toda la gente que me ha animado y acompañado durante estos años.

En especial agradecer a mi familia por todas las oportunidades que me han ofrecido, por su apoyo incondicional, por sus ánimos en los momentos más duros y las felicitaciones y celebraciones en los buenos. Gracias a mi madre, por estar ahí siempre, por su preocupación, sus consejos y por ser un pilar fundamental durante todos estos años de carrera. Por otro lado, agradecer a mi padre su apoyo continuo y por trasmitirme paz y seguridad en los momentos cruciales. Por último, pero no menos importante, gracias a mi hermana por ser siempre un ejemplo para mí, por su apoyo y su cariño.

Por último agradecer a todos mis amigos, tanto de la universidad como de fuera, que durante los momentos más duros han estado ayudándome y dándome ánimos. En especial gracias a mis compañeras y ya amigas de carrera Andrea, Celia, Ariana y Ada, por todos los buenos ratos y risas durante la carrera pero también por escucharme en los momentos de agobio.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

GLOSARIO

CAT: *Conditional Access Table*, Tabla de accesos condicionales: tabla con información sobre los accesos condicionales.

CC: *Continuity Counter*, Contador de Continuidad: campo de la cabecera MPEG-TS para contabilizar errores.

ES: *Elementary Stream*, flujo elemental de video o audio.

IP: *Internet Protocol*, Protocolo de Internet: protocolo de comunicación clasificado en la capa de red según el modelo OSI.

IPTV: *Internet Protocol Television,* Televisión sobre IP: se le denomina así a los sistemas de distribución de televisión por internet.

MPEG-TS: *Moving Picture Experts Group-Transport Stream*: Grupo de Expertos en Imagen en Movimiento- Flujo de Transporte: trama de transporte que contiene la secuencia de paquetes de audio, video y datos.

NIT: *Network Information Section,* Sección de información sobre la red: es una tabla que ofrece información sobre el estado de la red

PAT: *Program Association Table*, Tabla de asociación de programas: es una tabla con información sobre los distintos programas que contiene el flujo de trasporte.

PES: *Packetized Elementary Stream,* Flujo Elemental Paquetizado: formada por flujos elementales de audio y vídeo.

PID: *Packet Identification*, Identificación de Paquete: campo de identificación de los flujos.

PMT: *Program Map Table*, Tabla de información de programas: contiene información sobre todos los programas del flujo de transporte

RTSP: *Real Time Streaming Protocol*, Protocolo de *Streaming* de Tiempo Real: establece y controla uno o más flujos sincronizados de datos (audio y vídeo).

SDP: *Session Description Protocol*, Protocolo de Descripción de Sesión: describe sesiones de comunicación multimedia.

TCP: *Transmission Control Protocol,* Protocolo de Control de la Transmisión: protocolo de nivel de la capa de trasporte orientado a conexión y fiable.

TS: *Transport Stream*, Flujo de Transporte: protocolo de comunicación para para audio y vídeo.

UDP: *User Datagram Protocol*, Protocolo de Datagramas de Usuario: protocolo de nivel de la capa de trasporte basado en el intercambio de paquetes.

VoD: *Video on Demand*, Vídeo bajo Demanda: sistema de televisión que permite a los usuarios acceso a contenido Multimedia bajo petición.

1. Introducción

1.1. Motivación

En los últimos años, el modo de visualizar contenidos audiovisuales ha sufrido grandes cambios. En concreto, se ha visto modificado el método de difusión y distribución de servicios multimedia debido a la entrada de los sistemas IPTV (*Internet Protocol Television*, Televisión sobre IP). La principal característica de esta tecnología es que se trasmiten los contenidos a través de la red, reservando una parte del ancho de banda para la distribución de estos servicios. Este nuevo sistema ha permitido que muchos operadores de red hayan introducido en su oferta la posibilidad de contratar estos servicios y, en consecuencia, la opción de ver contenidos audiovisuales bajo demanda, tales como películas, documentales o series de televisión. Las redes IPTV utilizan dos tecnologías diferentes según el procedimiento por el cual se visualiza un contenido. Para la trasmisión de canales de televisión se utiliza una distribución *multicast* en la cual el contenido se trasmite solo una vez y se replica hasta llegar a todos los clientes suscritos a él. Sin embargo, en nuestro trabajo nos vamos a centrar en los servicios de VoD (*Video on Demand*, Vídeo bajo Demanda) que utilizan una distribución *unicast* y recurren al protocolo de señalización RTSP (*Real Time Streaming Protocol*, Protocolo de *Streaming* de Tiempo Real) para su trasmisión.

La característica principal de los servicios VoD es que los contenidos se trasmitirán únicamente cuando el usuario los solicite y él es el que tiene el control de pausar la trasmisión, reanudarla, rebobinarla y, ya por último, terminarla. Por este motivo, cada sesión y trasmisión tiene como destinatario un único usuario final. Todos estos mensajes de control entre el cliente y servidor serán los que se codifiquen mediante el protocolo RTSP.

Dada la relevancia que los servicios IPTV están adquiriendo hoy en día, cada vez es más importante disponer de herramientas de monitorización en tiempo real de estos sistemas. Con ellas, un proveedor de servicios de red podrá saber en tiempo real la calidad del servicio que está prestando.

Cuando un usuario solicita la reproducción de una película, el decodificador de la televisión o *Set-Top-Box* desempeñará el papel de cliente e interactuará con el servidor de contenidos multimedia a través del protocolo RTSP. Nuestro propósito en este trabajo ha sido la monitorización de estos servicios a través de un seguimiento de este protocolo. Además, si queremos tener un buen reporte sobre el desarrollo y la calidad de la sesión, será importante realizar también una monitorización de los flujos de vídeo que se trasmitirán por MPEG-TS (*MPEG-Transport Stream).*

1.2. Objetivos del TFG

El objetivo principal de nuestro TFG es desarrollar una herramienta que sea capaz de monitorizar los servicios de vídeo sobre redes a través del análisis de la señalización RTSP entre el cliente y el servidor multimedia. Para ello, realizamos un seguimiento de todos los mensajes intercambiados entre el cliente y el servidor multimedia y como resultado extraemos los datos más importantes de cada sesión. Uno de los propósitos principales del estudio de esta comunicación es identificar por dónde se va transmitir el flujo multimedia. Una vez detectados los puertos y direcciones por los que está enviando el servidor el vídeo al usuario, podemos capturar estos paquetes y realizar las medidas oportunas con respecto a la calidad en la trasmisión de los contenidos multimedia. De esta manera, dispondremos de una herramienta que reporta información de las incidencias que estén ocurriendo durante la trasmisión de los contenidos audiovisuales. Dentro de la información reportada se incluirán tanto errores en la señalización como pérdidas de paquetes en la trasmisión del flujo de vídeo.

En resumen podemos decir que los objetivos de este TFG son:

- Desarrollar una herramienta que sea capaz de monitorizar los servicios de vídeo sobre redes IP en tiempo real. Para ello, será importante que nuestra herramienta procese los paquetes lo más rápido posible.

- Reportar los datos extraídos durante la sesión, tales como errores en la señalización, la duración de las trasmisiones, las peticiones detectadas, la información relativa al trasporte del flujo multimedia …

- Identificar por dónde se está trasmitiendo el flujo multimedia y capturar aquellos paquetes correspondientes al vídeo.

- Realizar medidas sobre las pérdidas de paquetes en el flujo multimedia, de manera que periódicamente nuestro programa reporte al operador de red en tiempo real cuál es la calidad de la trasmisión. De este modo, si se produjera alguna incidencia, el proveedor podría identificarla y solventarla a tiempo.

1.3.Fases de realización del TFG

La realización de este TFG ha constado de cuatro fases principales:

- **Documentación**: En la primera fase se procedió al estudio de la tecnología que se utiliza para visionar vídeos bajo demanda mediante la señalización RTSP. Una vez obtenida una idea de la situación actual, se analizó en profundidad cómo funciona el protocolo de señalización RTSP y cuáles son los parámetros de utilidad para nuestro trabajo. Para poder ver cómo se utiliza este protocolo en un caso real se consiguió un archivo pcap de una transmisión real del proveedor Movistar, en la cual se utilizaba este protocolo.

- **Desarrollo**: Esta etapa es en la que se desarrolló el código y se hizo un análisis en mayor profundidad del problema que nos atañe. Se dividió principalmente en dos etapas:

    - Identificación y análisis de flujos RTSP: Después del estudio del funcionamiento del protocolo, se realizó un programa que analizó los flujos RTSP y detectó por qué puertos se estaba enviando los flujos multimedia en tiempo real. Una vez concluida la sesión, el programa genera un archivo con los datos más relevantes de ella y con los posibles fallos que se hayan podido producir.

    - Análisis de los flujos multimedia: Una vez detectado el puerto por el que se envía el flujo multimedia, se realizan periódicamente medidas de calidad sobre el vídeo en emisión.

- **Validación**: En esta esta etapa se realizan las pruebas necesarias para comprobar que nuestro programa funciona correctamente y poder resolver los problemas que hallamos detectado.
  Para poder validar nuestro programa se capturó el tráfico de Internet correspondiente a vídeos bajo demanda del proveedor de Movistar, obteniendo varias capturas. En la validación se analizan dos aspectos:

    - Errores en la señalización: Se valida que no haya problemas de conexión o pérdida de paquetes en la comunicación del flujo RTSP.

    - Pérdidas en la transmisión: Se introducen pérdidas controladas en nuestra captura de validación y se comprueba que son coherentes con las obtenidas por nuestro programa.

- **Redacción de la memoria**: Se redactó la memoria que describe el trabajo realizado y que, después de ser revisada por el tutor, se entrega la comisión de evaluación del trabajo de fin de grado.

A continuación, la Figura 1 muestra a través de un diagrama de Gantt la planificación temporal del trabajo. En ella, aparecen en color naranja las fases de realización y, para cada una de ellas, abajo y en azul las sub-fases que incluye.

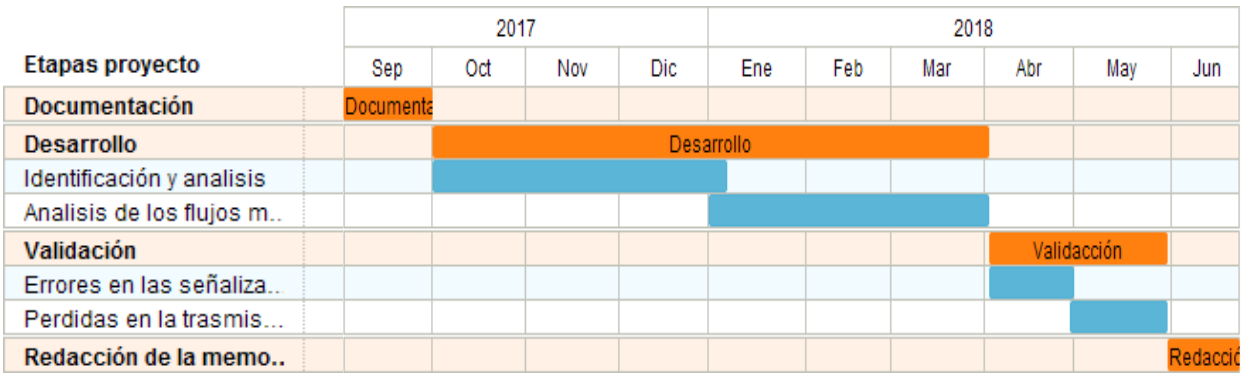

Figura 1. Diagrama de Gantt.

1.4. Estructura del documento

Esta memoria se ha divido en seis partes bien diferenciadas a través de las cuales se describe de una manera clara y organizada cómo se ha desarrollado el Trabajo de Fin de Grado y cuáles han sido los resultados obtenidos:

- **Introducción**: El documento comienza con una introducción que presenta en qué va a consistir el Trabajo de Fin de Grado y cuáles son los objetivos que se quieren alcanzar. Además, se explican las fases del proyecto y se introduce la estructura del mismo. Este primer capítulo es en el que nos encontramos ahora mismo.
- **Estado del arte**: En este segundo capítulo hablaremos sobre cómo se están desarrollando los servicios de vídeo sobre IP en la actualidad. Además, describiremos el protocolo RTSP, en el cual nos hemos basado para monitorizar los servicios de *streaming*. Por último, hablaremos sobre el protocolo de transporte MPEG-TS, en el cual nos hemos centrado para sacar las medidas de calidad de los flujos de vídeo.
- **Diseño**: Después de haber mostrado cómo funciona la tecnología en la que nos hemos basado realizaremos un análisis del problema a resolver y enumeraremos los requisitos que nuestro programa tendrá que cumplir. Además, organizaremos el diseño de nuestro programa en cuatro fases y utilizaremos un diagrama para explicar el diseño global de la herramienta.
- **Desarrollo**: En esta etapa explicaremos como hemos identificado los diferentes flujos de señalización RTSP y cuál ha sido la estructura del programa que hemos desarrollado para conseguirlo. Posteriormente explicaremos como hemos conseguido obtener medidas de calidad de los flujos de vídeo una vez detectados.
- **Integración, pruebas y resultados**: Una vez descrito el programa desarrollado detallaremos como hemos conseguido los archivos de validación (pcap) para comprobar que nuestro programa funciona y los casos en los que los hemos probado.
- **Conclusiones y trabajo futuro**: Por último, se mostrarán las conclusiones a las que hemos llegado y en qué aspectos se podría seguir trabajando en el futuro. Además, indicaremos qué asignaturas de la carrera han ayudado a sentar las bases y conceptos útiles para la realización de este trabajo.

2. Estado del arte

2.1 Introducción

En este segundo capítulo introduciremos el concepto de servicios sobre redes IP y daremos una visión general sobre el papel que están desempeñando en la actualidad. Además, hablaremos sobre el protocolo de señalización RTSP, que es en el que hemos basado nuestro trabajo de monitorización de los flujos de vídeo que se dan en una trasmisión de VoD.

Por último, describiremos el protocolo de comunicación MPEG-TS, que es el que se utilizó para enviar el flujo de vídeo y en el cual nos hemos centrado para obtener las medidas de calidad de la transmisión.

2.2 Servicios de vídeo sobre redes IP

IPTV es la denominación de los sistemas que proporcionan un servicio de televisión a través de Internet (sobre el protocolo IP) reservando una parte del ancho de banda a este servicio [1]. Esta tecnología en los últimos años ha revolucionado la manera de ver la televisión en los hogares. A diferencia del modelo de televisión por cable, al utilizar las redes IP, el cliente recibirá los contenidos solo cuando los solicite, en vez de que se estén transmitiendo de manera continua esperando a que el usuario se conecte. Es decir, nos vamos a un modelo de personalización de material multimedia parecido a lo que puede ser un "cine virtual" en el cual el cliente podrá seleccionar y ver los contenidos todas las veces que quiera. Este modelo de televisión será por suscripción o pago de manera que el material que podremos visualizar dependerá del tipo de suscripción que tengamos.

Dentro de los servicios de vídeo sobre redes IP tenemos que hacer distinciones entre dos usos de tecnologías diferentes:

- Tecnología *multicast*: En este caso la distribución de cada canal de TV se transmite una sola vez y llega a todos los clientes que estén suscritos al grupo de *multicast* asociado a él. La señal de televisión llegará replicándose por el camino.

- Tecnología *unicast*: Es la que se utiliza en el vídeo bajo demanda (VoD) puesto que en este caso cada transmisión dependerá únicamente del cliente que la solicite. Además, éste puede parar la transmisión, rebobinarla, avanzar o retroceder tantas veces como quiera.
  En nuestro trabajo nos centraremos en este tipo de estructura puesto que serán los vídeos bajo demanda los que nuestro programa monitorizará a través del protocolo RTSP.

2.3 Señalización RTSP

RTSP es un protocolo de trasmisión en tiempo real a nivel de aplicación de flujo de datos. Su función consiste en establecer una conexión entre un cliente y un servidor multimedia de forma que controla y sincroniza los flujos de vídeo y de audio de la sesión [2]. Se podría decir que, en sintaxis y en la manera de funcionar, es parecido al HTTP, aunque ambos protocolos se diferencian en aspectos importantes, tales como que RTSP necesita tener conocimiento periódico del estado de la conexión, o que los datos son transportados por un protocolo diferente.
A la hora de transmitirse RTSP es independiente del protocolo de transporte que vaya por debajo, puede ir tanto sobre TCP como UDP. Sin embargo, lo usual es encontrarlo sobre el primero. Para mantener un control sobre cada sesión asocia un identificador único a cada una de ellas.

En la Figura 2 se muestra un diagrama básico de una sesión RTSP.

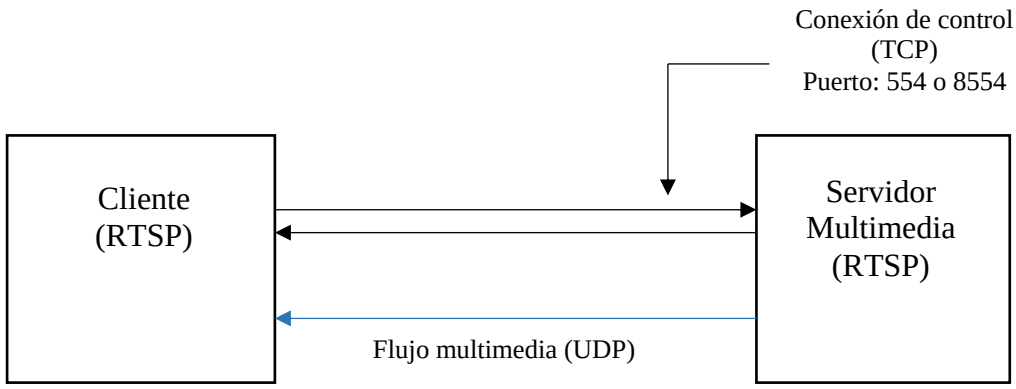

Figura 2. Diagrama básico de una sesión RTSP.

Describimos ahora como se establecería una sesión común de RTSP y cuál sería la cronología de las peticiones, representada en la Figura 3. En primer lugar, el cliente envía al servidor multimedia una petición en formato HTTP para establecer una conexión. Posteriormente, para abrir la sesión RTSP el cliente manda en primer lugar una petición OPTIONS al servidor, informándole de la configuración de la sesión. Si esta configuración es aceptada por el servidor, el cliente responderá con una petición DESCRIBE, que incluirá la descripción de la presentación que queremos visionar. En esta ocasión, el servidor contestará con los valores de inicialización necesarios para poder reproducirla. A continuación, el cliente manda una petición SETUP en la cual se fijan los puertos y las especificaciones de transporte (protocolos) por los que se van a transmitir el flujo de *streaming*. Por último, el cliente envía una petición PLAY, tras la cual servidor multimedia comienza a transmitir el flujo multimedia de la presentación pedida al puerto y con el protocolo de transporte establecidos. Durante la sesión, el cliente envía peticiones GET_PARAMETER periódicamente para saber que el servidor continúa activo. Por último, en el momento en que el cliente quiere finalizar la sesión, envía un TEARDOWN.

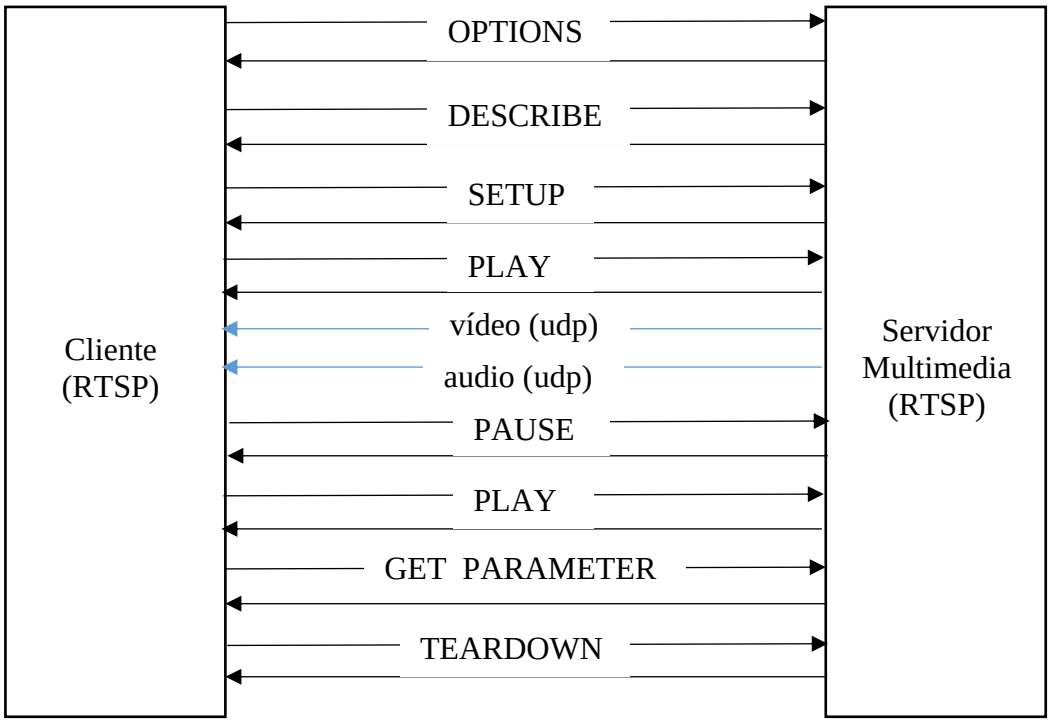

Figura 3. Diagrama de peticiones de una sesión RTSP.

A continuación, vamos a explicar con más detalle alguno de los métodos que nos van a ofrecer información importante que utilizaremos para monitorizar los flujos:

- *DESCRIBE*: Su función es la de inicializar la sesión. Para ello solicita una descripción al servidor de la presentación que queremos visualizar.

- *REPLY DESCRIBE*: Es la respuesta del servidor multimedia, que contendrá un mensaje SDP (*Sesion Description Protocol*, Protocolo de Descripción de Sesión) que proporciona todos los parámetros de inicialización del flujo multimedia. En concreto, contendrá información sobre el protocolo de transporte y el número y tipo de flujos que se van a enviar (audio y vídeo). En la Figura 4 se muestra un ejemplo de *REPLY DESCRIBE*.

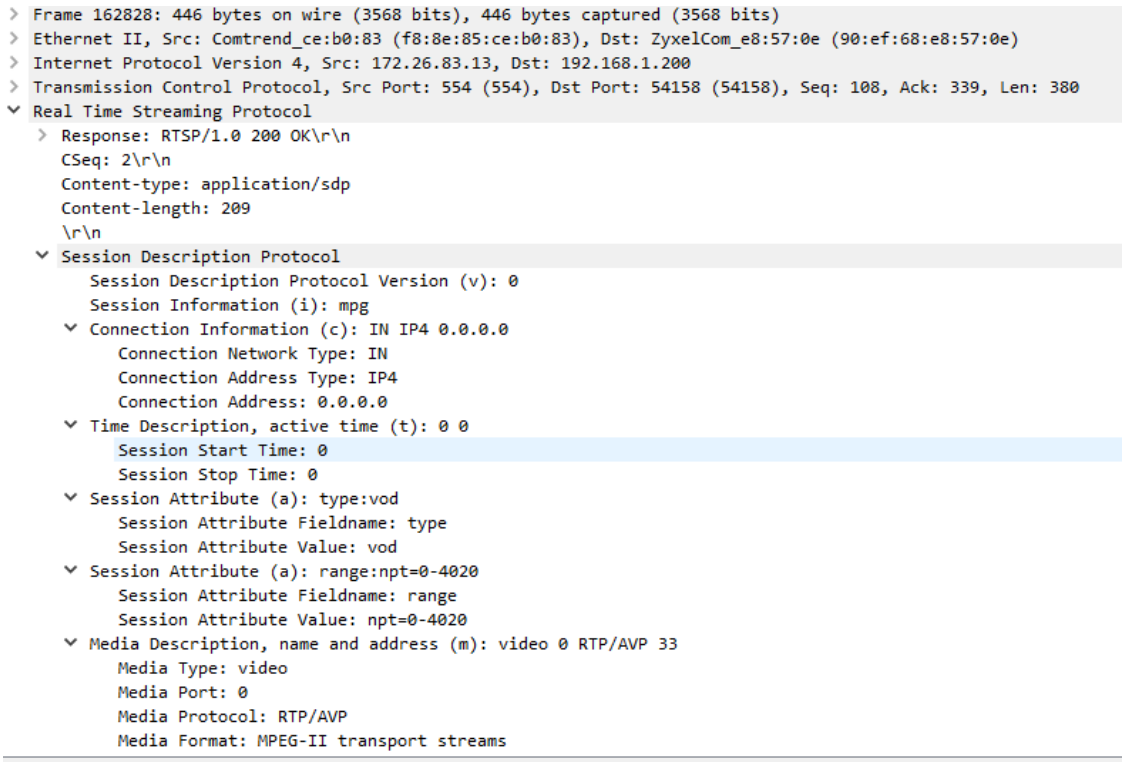

Figura 4. Ejemplo de respuesta a petición describe.

- *SETUP*: Este método es el que se encarga de acordar los parámetros del protocolo de transporte y los puertos por los que se va a enviar y recibir el flujo de audio y vídeo. En la Figura 5 se muestra un ejemplo de petición *SETUP*.7

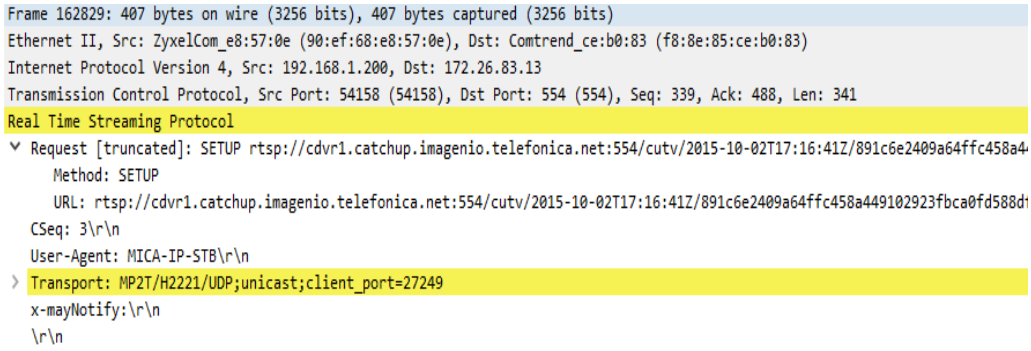

Figura 5. Ejemplo de petición SETUP.

- *REPLY SETUP*: Es la respuesta del servidor multimedia en la cual este último nos informa si acepta los protocolos de trasporte por los que se va a enviar el video a través del código de respuesta. Además en esta respuesta el servidor añadirá el puerto por el cual va a enviar el flujo multimedia. En la Figura 6 se muestra un ejemplo de la respuesta SETUP.

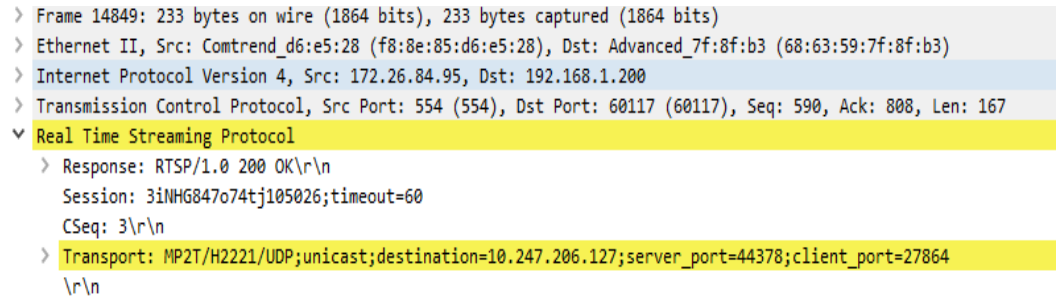

Figura 6. Ejemplo de respuesta a petición SETUP

- *PLAY*: Indica que el cliente ya está preparado y quiere empezar con la reproducción de la presentación solicitada. En esta petición es en la cual se va a fijar el identificador de sesión que se utilizara durante todo el tiempo de vida de esta misma. En la Figura 7 se muestra un ejemplo de la petición *PLAY*.

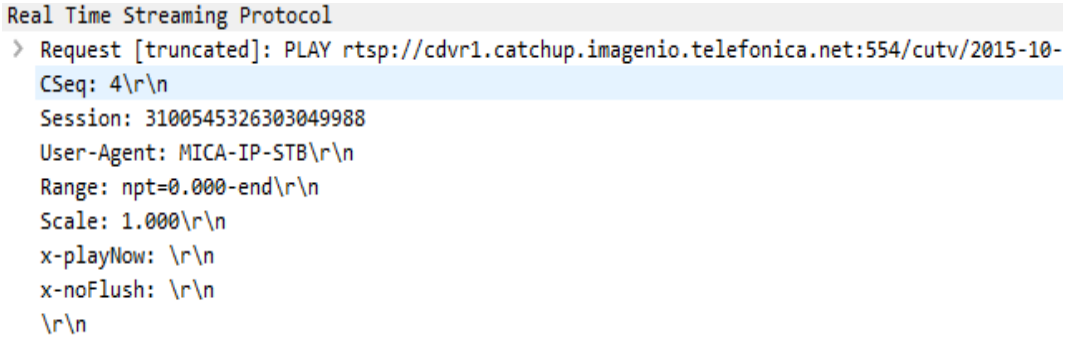

Figura 7. Ejemplo de petición PLAY.

– *TEARDOWN*: Indica que el cliente quiere finalizar la conexión con el servidor multimedia. Una vez que el servidor recibe esta petición deja de trasmitir los flujos de audio y vídeo, y se cierra la sesión. En la Figura 8 se muestra un ejemplo de la petición *TEARDOWN*.

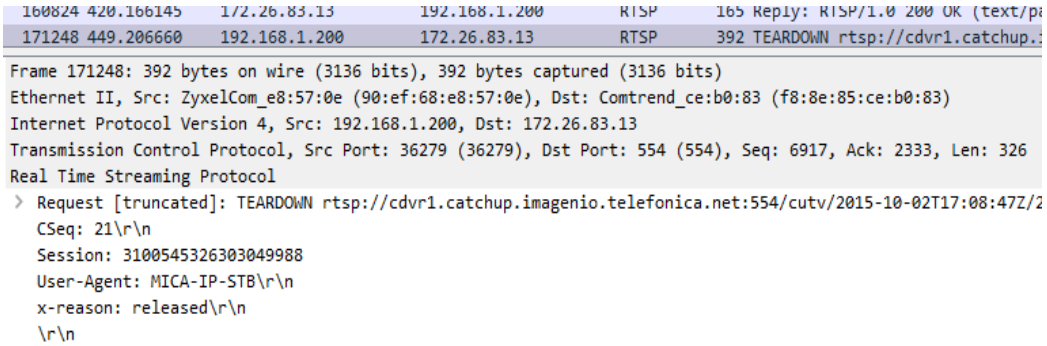

Figura 8**.** Ejemplo de petición TEARDOWN.

2.4 MPEG-TS

MPEG-TS es un estándar para transmitir los flujos de audio y vídeo. Estos flujos se comprimen de manera independiente, formando cada uno de ellos un *Elementary Stream* (ES). Los paquetes ES se agrupan posteriormente en paquetes PES, y por último, se encapsulan en un *Transport packet*, tal y como se muestra en la Figura 9. En el caso de una secuencia de vídeo, cada imagen se corresponde con un ES.
Cada *Transport packet* tiene siempre una longitud fija de 188 bytes y constará de una cabecera de 4 bytes, seguido de un campo de adaptación (opcional) y de la carga útil.

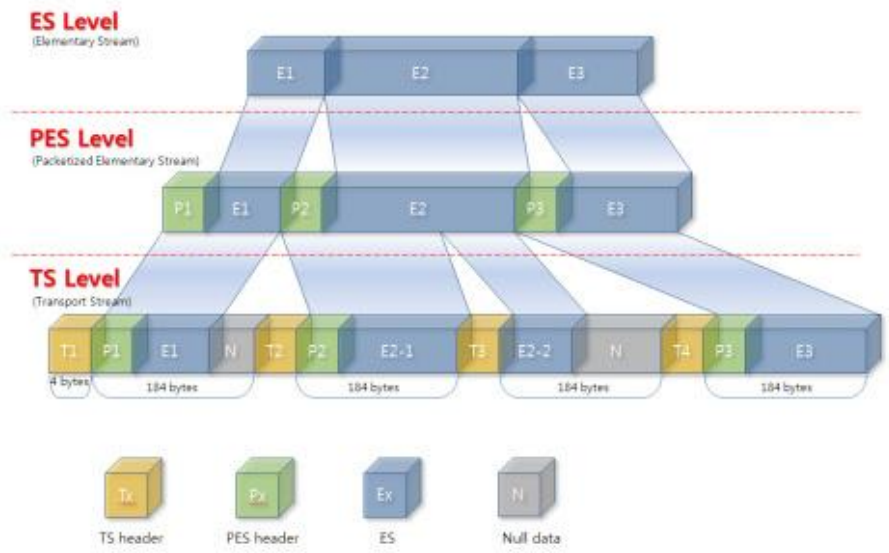

Figura 9. Encapsulación Transport Stream **[3]**.

Los paquetes de audio y vídeo (PES) se multiplexan juntos con otros datos de señalización y sincronización, con la intención de formar un solo *bitstream* llamado *Transport Stream*. No existen condiciones para determinar el orden en que los *Transport Packets* deben colocarse en el multiplexor, aunque sí, que debe respetarse el orden de los que pertenezcan a un mismo ES. Además, se añadirán paquetes de transporte nulos para mantener la tasa constante.

En la Figura 10 podemos ver un ejemplo sencillo de multiplexación TS. En ella podemos ver como el *Transport Stream* es el resultado de la multiplexación de diferentes flujos de paquetes entre los que nos podemos encontrar de audio, video, datos, servicio y los paquetes nulos para mantener la tasa que hemos mencionado anteriormente.

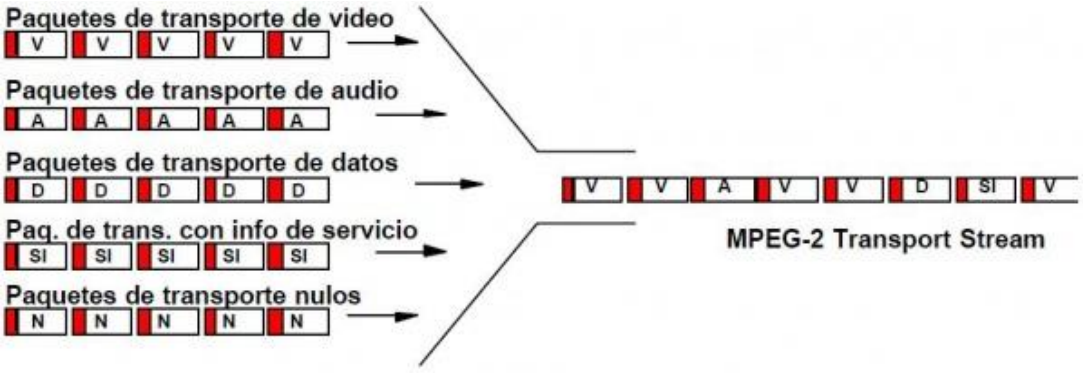

Figura 10. Ejemplo sencillo de multiplexación TS **[4]**.

La cabecera de un paquete TS va a constar de las siguientes partes (ver Figura 11):

- Byte de sincronización: Se utiliza para que el decodificador pueda sincronizarse adecuadamente.

- Indicador de error de transporte: Si está a 1 indica que hay un error en la transmisión.

- Indicador de arranque: Marca si en la cabecera hay un paquete PES.

- Prioridad de transporte: Si está a 1 indica que el paquete tiene prioridad.

- Identificador del paquete: Identifica qué tipo de *Elementary Stream* está encapsulado en ese TS. Hay 17 valores de PID que están reservados, pero el resto puede utilizarse por el multiplexor para identificar cada flujo de vídeo y audio. Este campo ocupará 13 bits.

- Control de cifrado: Indica si hay datos cifrados.

- Control de campo de adaptación:

    - 00: Reservado.
    - 01: No hay campo de adaptación, si hay carga útil.
    - 10: Hay campo de adaptación, no hay carga útil.
    - 11: Hay campo de adaptación y carga útil.

- Contador de continuidad (*continuity counter*): Sirve para saber si se ha producido una pérdida de algún paquete PES. Aumentará en 1 cada vez que se envíe un paquete de la misma fuente. Cuando el contador llega a 15 vuelve a comenzar en el 0. En el proyecto nos basamos en él para calcular las pérdidas del flujo de vídeo

- Campo de adaptación: Es opcional y los ocho primeros bits nos indican la longitud de este campo

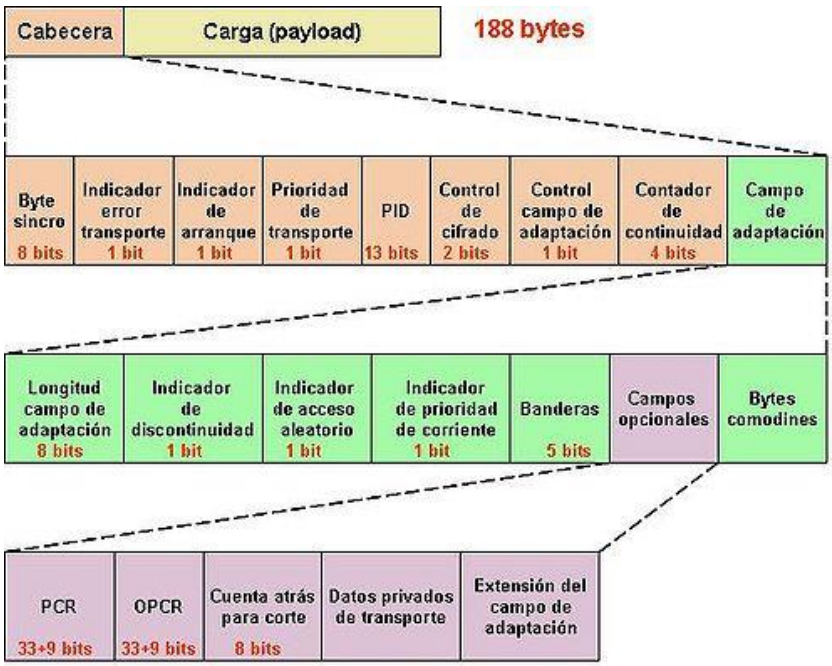

Figura 11. Cabecera MPEG-TS **[5]**

MPEG-TS define una serie de tablas de información de servicio que se deben incluir para poder decodificar los flujos de audio y vídeo de manera correcta en un *Transport Stream*. Para ello, MPEG-TS utiliza el *Program Specific Information* que le va a permitir no solo decodificar los flujos, sino también saber cómo están organizados estos mismos, dentro de la trama de trasporte. Esta información nos la van a proporcionar las tablas de información de servicio, las cuales viajan dentro de los flujos de trasporte formando *Elementary Streams* como ocurre con el video y el audio. Esto quiere decir, que cada tabla llevará asociado un identificador (PID) único en cada *Transport Stream*.

Es importante tener en cuenta que todos los paquetes que forman parte de un mismo Elementary Stream tienen el mismo PID. De esta manera, un decodificador busca en las tablas cual es el identificador para un ES y filtra a raíz de el para quedarse con el flujo de video, audio o datos.

Tablas obligatorias:

- PAT (*Program Association Table*, Tabla de asociación de programas): Nos informa sobre los diferentes programas que va a contener el *Transport Stream*. Su ID siempre va a ser el 00. Además, a partir de ella podemos saber cuál es el PID de la tabla PMT. Por cada *Transport Stream* habrá una única tabla PAT que se mandará al comienzo de este mismo.

- PMT (*Program Map Table*, Tabla de información de programas): Proporciona información sobre cada *Elementary Stream* del *Transport Stream*: el PID en el que viaje la trama fundamental, el tipo de trama que es (audio, vídeo) y descriptores asociados a ella. De esta manera el decodificador sabrá en que paquete va cada *Elementary Stream* y como decodificarlos.

- CAT *(Conditional Access Table,* Tabla de accesos condicionales): Tiene información sobre el acceso condicional, su PID es el 1. Es obligatoria cuando en el *Transport Stream* existe algún programa que se encuentra codificado.

- NIT *(Network Information Section,* Sección de información sobre la red): Esta tabla es opcional y nos proporciona información de red. Su PID lo sabremos a extrayéndolo a través de la tabla PAT. Si en nuestro Transport Stream existe esta tabla lo localizamos porque es el programa 0 de la tabla PAT

2.5 MPEG-TS sobre IP

MPEG-TS puede transportarse de dos maneras diferentes:

- Sobre UDP: En este caso, tras la cabecera de UDP nos encontraremos directamente con la cabecera del primer paquete del *Transport Stream* (TS). En cada paquete de UDP irán 7 paquetes de TS. Esta va ser la situación que nos encontraremos en nuestro trabajo, puesto que en los casos estudiados es el método que emplean para trasmitir los flujos multimedia. Para poder realizar medidas sobre calidad habrá que proceder a la decodificación del MPEG-TS puesto que la cabecera UDP no nos ofrece información de control de pérdidas de paquetes.

- Utilizando RTP: Aquí, el tipo de carga útil de la cabecera RTP (33) nos indicará que a continuación nos encontramos con una cabecera MPEG-TS. La cabecera RTP cuenta con un número de secuencia que servirá para saber si ha habido pérdidas de algún paquete durante la transmisión. En la Figura 12 podemos ver una representación de los campos de la cabecera RTP. En ella el número de secuencia está indicado como *Sequence number*. Tras esta cabecera nos encontraremos con la cabecera del primer paquete PES del *Transport Stream*(TS).

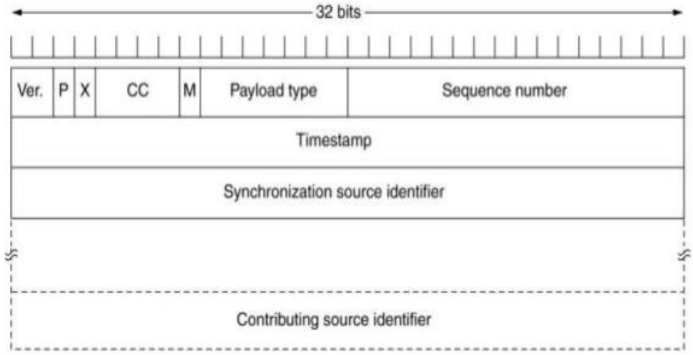

Figura 12. Cabecera RTP **[6]**

2.6 Conclusión

En este capítulo hemos explicado las diferentes tecnologías en las que nos hemos basado en el trabajo para el desarrollo de nuestro programa. Primero, hemos explicado cómo en los últimos años han tomado cada vez más relevancia los servicios de vídeo sobre IP y, centrándonos en nuestro caso, los de VoD (vídeo bajo demanda). Hemos continuado describiendo cómo funciona la señalización RTSP, y cómo nos permite identificar por dónde se está transmitiendo el flujo multimedia. Por último, hemos repasado cómo funciona el estándar MPEG-TS, a través del cual se trasporta nuestro flujo de vídeo. En particular, hemos expuesto los parámetros que nos van a permitir identificar qué paquetes pertenecen a nuestro flujo de vídeo y cómo comprobar si ha habido pérdidas de paquetes. En los capítulos siguientes utilizaremos toda esta información para diseñar y desarrollar el programa

3. Diseño

3.1. Introducción

Como ya hemos visto, el reciente incremento de sistemas que distribuyen material multimedia sobre redes IP, requiere de nuevas herramientas que monitoricen y controlen la calidad de estos servicios. En este capítulo, nos detendremos en analizar los requisitos que debe cumplir nuestro programa para ofrecer unos buenos resultados, a la hora de reportar y ofrecer información en tiempo real sobre la calidad de estos servicios.

Una vez estudiados estos requisitos, tendremos una idea más global sobre el diseño que debe tener nuestro programa y de los objetivos que debemos alcanzar a la hora de desarrollar el proyecto.

Además, como este Trabajo de Fin de Grado consiste en el desarrollo de un sistema de monitorización en tiempo real, una característica fundamental de su diseño debe ser que su coste computacional no sea muy grande. Por otro lado, nuestro proyecto deberá ser capaz de procesar y sincronizar diferentes flujos y, por tanto, ofrecer una solución estable y fiable será de gran ayuda.

3.2. Análisis de requisitos

A continuación, vamos a describir los requisitos que nuestro programa debe cumplir. La exposición se subdivide en dos partes, requisitos funcionales y no funcionales con el fin de conseguir un mejor análisis del problema.

3.2.1 Requisitos funcionales

En este primer apartado, vamos a realizar un estudio de los requerimientos funcionales que deberá llevar a cabo nuestro programa, en otras palabras, las funcionalidades y objetivos a los que debemos dar solución. Esta sección se subdivide en varios módulos diferenciados por su función dentro del programa.

En primer lugar, el diseño del código se puede separar en cuatro módulos, ordenados cronológicamente en el esquema Figura 13 según se han ido desarrollando:

Figura 13. Diagrama del diseño del programa.

1. Identificación de tráfico RTSP: Esta etapa es la que se encarga de analizar el tráfico de internet e identificar a través del puerto de origen y destino los paquetes que vayan sobre RTSP.

2. Análisis de tráfico RTSP: Este módulo continuará el trabajo del anterior realizando un seguimiento de cada sesión RTSP detectada. Analizando las diferentes peticiones entre cliente y servidor podremos averiguar los puertos por los que se está enviando y recibiendo el flujo multimedia. Además, extraeremos los datos más importantes de cada una de las conexiones establecidas.

3. Identificación de flujos de vídeo: A raíz de los datos extraídos en el módulo anterior, identificaremos los paquetes por los cuales se está trasmitiendo el flujo multimedia y los filtraremos hasta quedarnos solo con los que pertenezcan al flujo de vídeo que nos interesa.

4. Medidas de calidad: Por último, estimaremos si ha habido alguna pérdida de paquetes.

A continuación, expondremos los requisitos de cada uno de los módulos:

1. Identificación de tráfico RTSP:

    - 1: Analizar el tráfico a través de un archivo que se le pasará por argumento al programa.
    - 2: Filtrar el tráfico de internet de forma que seleccionemos solo los paquetes RTSP.

2. Análisis de tráfico RTSP:

    - 3: Realizar un seguimiento de la secuencia de peticiones de cada sesión extrayendo los datos necesarios.
    - 4: Conseguir que nuestro programa cuente con una estructura óptima para ser capaz de analizar distintas sesiones que se estén recibiendo intercaladamente en tiempo real sin que haya ningún fallo
    - 5: Volcar en un archivo de los datos más importantes de cada sesión.

3. Identificación de flujos de vídeo:

    - 6: Analizar los paquetes de MPEG-TS de forma que filtremos por los que tengan el PID correspondiente al vídeo que queremos extraer.

4. Medidas de calidad:

    - 7: Una vez detectado el flujo de vídeo comprobaremos si ha habido alguna pérdida de paquetes.

Finalmente, tendremos un último módulo que consistirá en la obtención de archivos de validación. Estos últimos nos serán de gran utilidad durante el desarrollo de nuestro programa.

5. Obtención de archivos de validación: Se realizan varias pruebas utilizando el reproductor multimedia VLC como servidor y como cliente y las trazas se capturan a través de Wireshark. Este sistema nos ayuda a modelar los diferentes escenarios que nos podemos encontrar. Para obtener una representación más fiel a la realidad se capturará en un hogar la trasmisión entre el decodificador y el servidor del proveedor de movistar.

A continuación, mostraremos un esquema del diseño global del programa (ver Figura 14). A nuestra herramienta se le pasará un fichero de red como argumento y comenzará a ejecutarse nuestro primer módulo de identificación de tráfico RTSP. Tras él, se iniciará el de análisis que seguirá el trascurso de la sesión. Una vez que concluya la sesión se genera un fichero de los datos extraídos. Por otro lado, una vez que el módulo de análisis RTSP haya recibido una petición PLAY, comenzará a ejecutarse el módulo de identificación de flujos multimedia y las medidas de calidad correspondientes que se reportarán cada 10 segundos en un fichero.

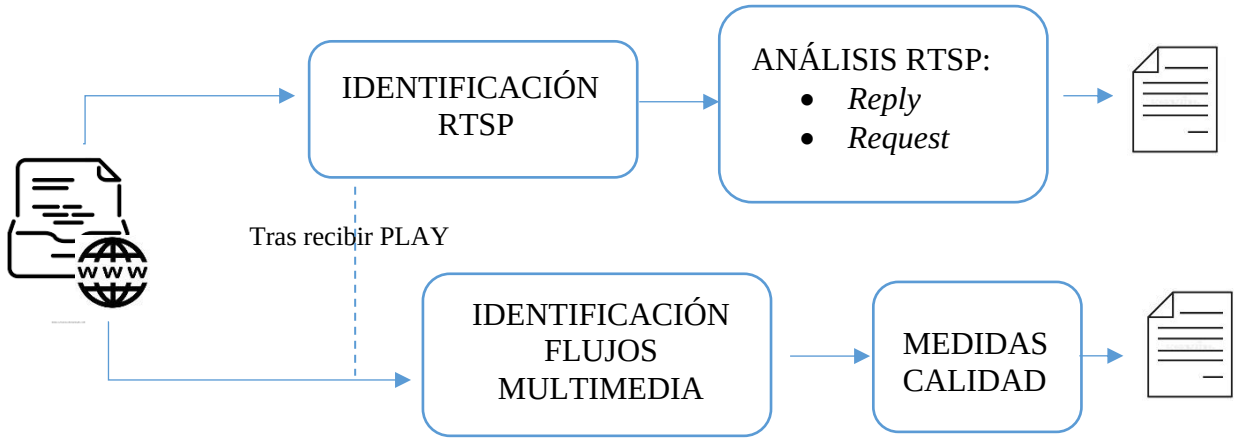

Figura 14. Diseño global del programa

3.2.2Requisitos no funcionales

En esta ocasión, nos centraremos en otro tipo requisitos técnicos que, aunque no son funcionales, son necesarios para que nuestro programa cumpla con sus requerimientos de una manera óptima.

- Estabilidad: Nuestra herramienta deberá ser muy estable, ya que al estar en continuo funcionamiento, un error repercutiría de forma grave en el resto del programa.

- Fiabilidad: El programa debe aportar datos fiables de forma que la información extraída sea un reflejo de lo que está sucediendo en la sesión.

- Portabilidad: La aplicación debe poderse ejecutarse en distintos dispositivos sin que ello suponga un problema. En nuestro caso el código estará escrito en lenguaje c de programación para que se pueda ejecutar en cualquier terminal de Linux

- Rendimiento: Será imprescindible que nuestro proyecto no tenga un coste computacional muy alto, puesto que al ejecutarse el programa en tiempo real será imprescindible que tenga un rendimiento elevado.

- Usabilidad: Es importante que nuestro programa sea sencillo de ejecutar y de interpretar. De la misma manera, el fichero con los datos extraídos deberá estar en un formato adecuado, para que posteriormente pueda ser procesado fácilmente por otro programa.

3.4 Conclusión

En este capítulo, hemos repasado el diseño que debe tener nuestra herramienta y, para ello, hemos hecho un estudio exhaustivo de las metas y los requisitos que nuestro programa debe alcanzar y cumplir. En resumen, podemos decir que nuestra aplicación debe cumplir con un diseño estable y de alto rendimiento que nos permita monitorizar el tráfico en tiempo real, aplicando a la entrada de tráfico de red los filtros pertinentes en cada caso. Por otro lado, debe tener un uso sencillo y de fácil comprensión puesto que la finalidad de nuestro programa es que las operadoras puedan utilizarlo para tener conocimiento del servicio que están ofreciendo en tiempo real. En el capítulo siguiente explicaremos como hemos desarrollado cada una de las fases del diseño.

4. Desarrollo

4.1 Introducción

Este capítulo es una continuación del apartado anterior y en él explicamos de forma más detallada y técnica cómo hemos realizado el programa. Para ello, utilizamos las etapas de diseño que hemos definido en el apartado anterior (ver Figura 13).
Una vez se haya explicado cada módulo, se describirá cuál ha sido la morfología y las estructuras de datos que hemos utilizado para llevar a cabo la sincronización de todos los módulos de nuestro código. Por último, hablaremos de la extracción de datos a un fichero de salida.

4.2 Identificación de tráfico RTSP

Esta primera parte del programa consiste en la identificación de los paquetes que usen la señalización RTSP. Para explicarla indicaremos como hemos ido filtrando a través de las capas de red, hasta llegar a la capa de aplicación que es en la que se encuentra nuestro protocolo (ver Figura 15). En este caso, sabremos si nos encontramos ante un paquete RTSP, fijándonos en el puerto del que proviene. En primer lugar, el programa pedirá como argumento de entrada un fichero de una captura de internet. Nótese, que si nuestro programa se implementara en la realidad debería cambiarse esta modalidad por otra que capturara directamente de la red.

A continuación, explicaremos la disección que realizaremos a cada uno de los paquetes que procesamos. Incluye dos pasos:

- Detección de paquete RTSP: Nada más llegar un paquete de la red, extraemos la cabecera Ethernet y nos fijamos en si el tipo Ethernet concuerda con el asignado al protocolo IP. Una vez que identificamos un paquete IP, nos quedamos con su protocolo y las direcciones de origen y destino. RTSP puede ir sobre dos protocolos de trasporte diferentes: TCP y UDP. Lo habitual es que vaya sobre TCP puesto que es un protocolo orientado a conexión. En cualquier caso, tanto si va en TCP o en UDP, debemos quedarnos con el puerto origen y el puerto destino de cada paquete. Para saber si nos encontramos ante un paquete RTSP debemos fijarnos si el puerto de origen o destino es el 554 (puerto por defecto) o el 8554. En ese caso, ya podemos asegurar que estamos ante una petición del cliente (puerto destino 554) o ante una respuesta del servidor (puerto origen 554).

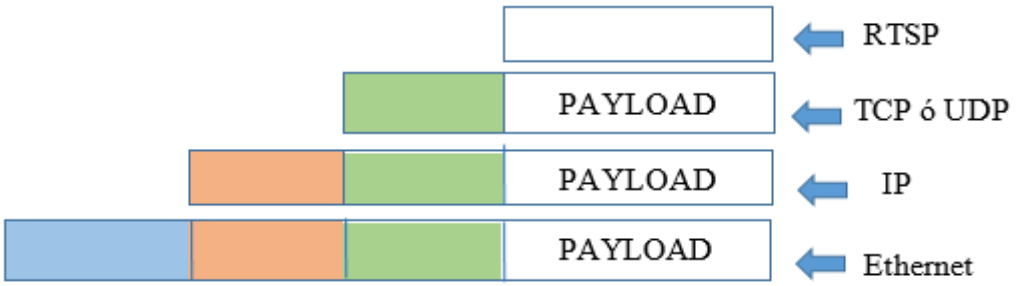

Figura 15. Esquema de encapsulado RTSP.

- Extracción de datos RTSP: En esta segunda parte, extraeremos la longitud de la cabecera TCP, aplicaremos una máscara y avanzaremos lo suficiente para apuntar a los datos dónde irá ya la información relativa al protocolo RTSP.

Una vez que concluya este módulo, nuestro programa debe disponer de la siguiente información: IP origen, IP destino, puerto origen, puerto destino, puntero a los datos (RTSP)

4.3 Análisis de tráfico RTSP

En esta segunda parte, procedemos al análisis de los datos RTSP. En concreto, en este apartado analizaremos los datos de cada petición y la respuesta entre el servidor y el cliente, de manera que tendremos un seguimiento total del trascurso de cada sesión establecida. Al tener que realizar una monitorización de cada sesión, podremos averiguar en qué momento se empieza a enviar los flujos multimedia y por qué puertos se envían.

Lo primero que hacemos una vez detectado un paquete RTSP es clasificarlo en petición del cliente o en respuesta del servidor. Posteriormente, consultamos si el paquete recibido pertenece a una sesión ya empezada o si es una nueva. En el caso de que forme parte de una sesión ya establecida, se cargaría la información ya recogida de esa sesión. Por otro lado, si nos encontramos ante una sesión nueva la cargaríamos en la tabla.

En este momento, comenzaríamos con el análisis de la sesión. Para poder explicar de manera más clara como se ha desarrollado esta etapa seguiremos el transcurso básico de una sesión normal, parándonos a explicar cómo procesamos cada petición (ver Figura 16).

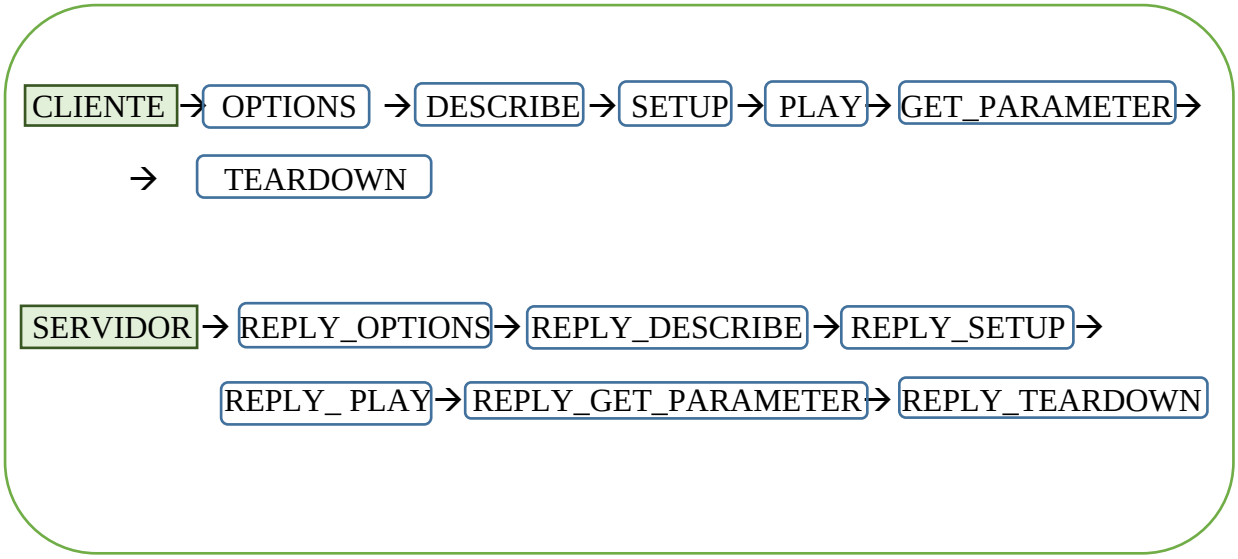

Figura 16.Esquema básico de sesión RTSP (Cliente, Servidor)

Cabe destacar que para todas las peticiones y las respuestas procesadas en cada sesión extraeremos: a) el instante en el que se ha recibido, b) si ha habido algún error, c) el número de secuencia y d) el método al que corresponde. De esta manera, al finalizar la sesión tendremos un resumen en el que expondremos la secuencia de peticiones que ha habido, el instante en el que se han recibido y los errores detectados.

Procesamiento de las **peticiones** entre cliente y servidor:

1. OPTIONS: Esta petición nos indica que el cliente quiere establecer una conexión con el servidor. En este momento se creará una nueva posición en la tabla de sesiones y se reservará memoria para una estructura donde se guardarán todos los datos relativos con la sesión.
   Como información útil extraeremos la lista de métodos que acepta el cliente y el número de secuencia de la petición.

2. DESCRIBE: Se encarga de solicitar la presentación al servidor y pedir los valores necesarios para inicializarla. Guardaremos la URL de la presentación pedida en la estructura de datos asignada a la sesión.

3. SETUP: El cliente especifica los protocolos de trasporte y el puerto origen al cual el servidor debe enviar los flujos multimedia. La información que conseguimos a través de esta petición es muy importante, puesto que nos va a servir para saber identificar cómo y por donde se está enviando el vídeo y así poder analizarlo posteriormente.

4. PLAY: Este mensaje es el que nos indica que el cliente ya está listo para empezar a recibir la presentación y reproducirla. A partir de este momento, nuestro módulo de identificación de flujos multimedia se pondrá en marcha.

5. PAUSE: A través de esta petición el cliente solicita que se pare de enviar el vídeo hasta que se reenvíe un PLAY.

6. GET_PARAMETER: El cliente envía estas peticiones periódicas para saber si el servidor sigue activo.

7. TEARDOWN: El cliente pide que se finalice la conexión. Una vez que procesamos una petición de cierre, nuestro programa se queda en espera de recibir la respuesta del servidor y cerrar la sesión.

A la hora de analizar las respuestas del servidor nos fijamos en los códigos de respuesta que nos documentan si hay algún tipo de error. Esta información se encuentra en la primera línea del mensaje cuyo formato será el siguiente: RTSP-Versión status-code \r\n

Procesamiento de las **respuestas** del servidor:

1. REPLY OPTIONS: El servidor contestará indicando si ha habido algún error y mostrando la lista de métodos que admite.

2. REPLY DESCRIBE: En la respuesta del servidor a la petición al DESCRIBE se obtienen los valores de inicialización de la presentación. En otras palabras, esta petición contará con un mensaje SDP que nos proporcionará información importante relativa al flujo multimedia. En concreto, obtendremos el número de flujos (vídeo y audio) y los códec de cada uno de ellos.

    - Reensamblaje TCP:

      Como ya dijimos anteriormente, RTSP normalmente irá sobre el protocolo de transporte TCP. Observando nuestras trazas vimos que los paquetes SDP a menudo iban fragmentados, a nivel de TCP, en varios paquetes. Para detectar estos casos especiales, comparamos la longitud del paquete TCP, que nos marca la cabecera, con el número de bytes que nos indica la respuesta RTSP. En el caso de que el número de bytes del paquete TCP es menor que lo que ocupa el mensaje SDP, procesamos el siguiente paquete.

    - Procesado del mensaje SDP:

      Como ya hemos visto, la respuesta a la petición describe requiere de un procesamiento distinto. En parte, por el posible reensamblaje a nivel de TCP mencionado anteriormente, y por el tipo de estructura del mensaje.
      En primer lugar, tenemos que contar con que el mensaje puede contar con información de un solo flujo o de varios (audio, video). En el caso de un solo flujo, buscaremos en nuestra petición la descripción multimedia que vendrá introducida por los siguientes caracteres (m):
      Un ejemplo de una descripción multimedia sería el siguiente:
      Media Description, name and address (m): video 0 RTP/AVP 33

En este caso podemos ver que se nos indica: el tipo de flujo multimedia, el protocolo encapsulación y el formato. El último número, 33 en este caso, nos indica que va codificado en MPEG-II-Transport Stream.
Si en nuestro mensaje vinieran información de varios flujos seguiríamos procesando la respuesta hasta obtener la información multimedia de todos.
Se pudo comprobar durante el desarrollo del programa que, en algunos casos, en la petición describe se especifica que el flujo va encapsulado de una cierta manera y posteriormente se comprobó que en los mensajes SETUP se establece unos protocolos de encapsulación diferentes.

3. REPLY SETUP: En esta respuesta se indicará la dirección desde la que se mandará el flujo multimedia y el puerto. Además, se indicará el protocolo de trasporte.

4. REPLY PLAY: Tras recibir la contestación de que no hay ningún error, se empieza a trasmitir el vídeo. Además, se introduce en la tabla de flujos de vídeo una nueva entrada con el número de identificación de la sesión RTSP.

5. REPLY PAUSE: Al contestar a la petición de pausa, se para la trasmisión del vídeo hasta que se reciba una nueva petición PLAY

6. REPLY GET_PARAMETER: Es la respuesta del servidor proporcionando datos sobre el estado de la sesión. Se utiliza para saber si el servidor sigue activo.

7. REPLY TEARDOWN: Es la contestación por parte del servidor a la petición del cierre de sesión. Una vez enviada, el servidor deja de trasmitir el flujo multimedia. Al recibirla, nuestro programa escribe en un fichero los datos más relevantes de nuestra sesión, y elimina la sesión de nuestra tabla de sesiones para liberar memoria. Además, hay que borrar la entrada del flujo de vídeo correspondiente en la tabla de flujos y escribir el fichero correspondiente con las medidas (explicado en los siguientes módulos).

    - <u>Ausencia de respuesta a TEARDOWN:</u>

      Al hacer pruebas con nuestro programa, se encontró que en algunas trazas la respuesta a la petición de TEARDOWN no existía, aunque el cliente sí que detectaba el cierre de sesión, y dejaba de enviar peticiones. Como solución creamos una función que, una vez detectado el TEARDOWN, mirará si en los siguientes mensajes TCP se mostraba que había un fin de sesión. Para ello, se hizo un seguimiento (ver Figura 17) de las banderas de FIN y ACK de la cabecera TCP. En primer lugar, el cliente envía un paquete TCP con el flanco de FIN activo y espera a que el servidor le conteste con un ACK. En este momento el servidor contesta con otro mensaje FIN y tras un ACK del cliente la sesión se da por terminada.

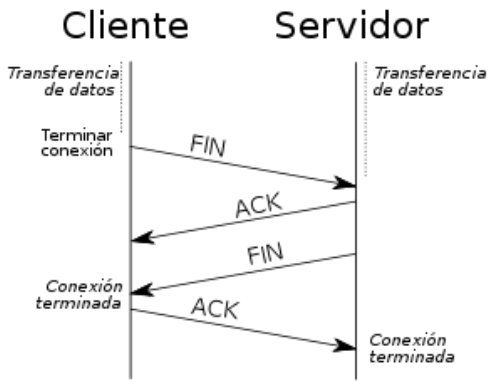

Figura 17 Ejemplo cierre de sesión **[7]**

Parseado de peticiones entre servidor y cliente:

A continuación, vamos a explicar el método empleado a la hora de parsear los mensajes que se intercambian cliente y servidor. Hay que tener en cuenta que dado que RTSP es un protocolo de la capa de aplicación no podemos tener un esquema bit a bit sobre en qué lugar está la información que buscamos como por ejemplo ocurre con los protocolos de trasporte TCP o UDP.
Si nos fijamos en la IETF de RTSP [2] podemos ver el formato que deberían seguir las peticiones y las respuestas entre el cliente y el servidor. Basándonos en esta estructura desarrollamos nuestro código para poder procesar los mensajes. Mientras se desarrolló el código se pudo comprobar que no siempre los servidores y el cliente siguen el mismo formato de peticiones. En ciertos casos se pudo detectar que en función del servidor multimedia utilizado, el orden de los diferentes campos dentro de los mensajes e incluso a veces el formato variaba (variaciones en espacios, mayúsculas, puntos…).
La obtención y detección de los datos que necesitamos se realizó a través de dos funciones. En primer lugar se usó strstr que localiza una cadena de caracteres en un texto. De esta manera podemos detectar el lugar donde se encuentra la información que queremos obtener. Una vez obtenido el puntero al comienzo de la cadena, usamos strtok para fragmentar la cadena y extraer únicamente el campo que necesitamos. Al utilizar la primera función strstr solventamos el problema de que no siempre nos envíen las peticiones y respuestas de la misma manera y orden.

4.3 Identificación de flujos de vídeo

Este tercer módulo comienza a ejecutarse cuando el servidor multimedia responde a la petición PLAY y, en consecuencia, empieza a trasmitirse el vídeo. Es decir, a partir de este momento el vídeo se está trasmitiendo desde el servidor hasta el cliente por los puertos que se han establecido previamente durante la sesión.
Mientras que la señalización RTSP va casi siempre sobre el protocolo de trasporte TCP, los flujos multimedia utilizaran el protocolo UDP, que no está orientado a conexión, e introduce menos retardos.

Fases de la identificación de vídeo:

- Fase 1: Cada paquete que llega sobre el protocolo UDP se filtra por la dirección IP de origen, IP destino, puerto origen y puerto destino que hemos hallado en el módulo de análisis de trafico RTSP correspondiente a cada sesión. Estos datos los hemos captado a través de las peticiones de SETUP y DESCRIBE en el módulo anterior.

- Fase 2: A través del identificador de sesión accedemos a la estructura de datos correspondiente al flujo multimedia. En este momento diseccionamos la cabecera UDP y vemos si el vídeo viene sobre RTP o directamente sobre MPEG-TS. En este trabajo, nos centraremos en la segunda opción. Si por el contrario, se usara RTP, las pérdidas de paquetes se contarían a través del número de secuencia.

- Fase 3: Avanzamos lo que nos queda de cabecera UDP y apuntamos directamente a los datos que serán el comienzo de la cabecera MPEG-TS. En este momento, sabemos que cada *Transport Stream* tiene hasta siete 7 paquetes TS de 188 bytes cada uno (ver Figura 9). Para poder identificar cuáles de todos los paquetes TS corresponde a nuestro flujo de vídeo, localizamos en primer lugar la tabla PAT y, a raíz de ella, la tabla PMT. Estas dos tablas nos indican los programas del *Transport Stream* y la información sobre los *Elementary Stream* que lo forman. De esta manera. tendremos acceso al PID del flujo de vídeo que queremos analizar.

- Fase 4: En esta última etapa, detectamos los flujos de vídeo que en nuestro caso irán sobre el códec H.264. Una vez obtenido el PID de los *Elementary Stream,* correspondientes a nuestro vídeo, procedemos a filtrar a través de la cabecera de MPEG-TS todos los paquetes. Ahora solo queda pasarle esta información al último módulo de diseño, el de medidas de calidad.

4.4 Medidas de calidad

Este módulo se encarga de obtener si ha habido alguna pérdida de paquete durante la trasmisión del flujo de vídeo y, en su caso, calcula el porcentaje de pérdidas que ha habido. Para poder hallar si ha habido pérdidas hemos ideado un modelo a base de realizar un seguimiento de los *Contunity Counter* que es un campo de la cabecera MPEG-TS. Este campo se incrementa en uno cada vez que se envía un *Elementary Stream* de la misma fuente. Es decir, todos los TS pertenecientes a nuestro flujo de vídeo tienen asignado el mismo PID y su campo CC (*Contunity Counter)* aumenta en una unidad cada vez que se codifica un nuevo paquete.

En nuestro trabajo realizamos dos medidas diferentes, una de los paquetes perdidos y otra de los paquetes TS pertenecientes a nuestro flujo que se han extraviado

Una vez que nuestro programa detecta que ha habido una discontinuidad en el CC, calcula de cuantas unidades ha sido la pérdida. En este momento, ya sabemos que como mínimo se ha perdido un paquete en la trasmisión. Según el valor de la diferencia entre el CC del anterior paquete y el nuevo, podemos hacer una estimación de cuantos paquetes no hemos detectado.

Estas medidas calculadas las reportaremos o de manera global al finalizar la sesión que monitorizamos o periódicamente cada cierto tiempo para tener un control en tiempo real sobre lo que sucede en la trasmisión del video. En este segundo caso el contador de paquetes perdidos se pondrá a cero cada vez que se saquen las estadísticas.

4.5 Datos extraídos

Por último, una vez que el programa detecta que hay una respuesta a la petición de TEARDOWN, se procede a la escritura de un fichero con los datos más importantes de la sesión. Este fichero llamado sesión.txt contendrá los siguientes campos:

- ID Sesión: Número de identificación único de cada sesión
- IP Cliente: Dirección IP del cliente que solicita el contenido audiovisual.
- IP Servidor: Dirección IP del servidor multimedia
- Puerto origen: Puerto del cliente en el cual recibe el flujo multimedia
- Puerto destino: Puerto del servidor por el que envía el vídeo
- Protocolo transporte RTSP: Protocolo de trasporte de la comunicación RTSP
- Duración: Duración desde que se inicia la conexión hasta que se termina
- Distribución: *Unica*st o *Multicast*
- Cliente: Nombre del cliente
- Servidor: Nombre del servidor multimedia
- Lista de métodos admitidos: Lista de métodos que admite el cliente
- Cronología de métodos de la sesión: El orden de todos los métodos enviados por el cliente al servidor.
- Tiempos de peticiones: El instante de tiempo en el que se envían todas las peticiones del cliente.
- Tiempos de respuesta : Instante de tiempo en el que el servidor contesta al cliente
- Url de la presentación: dirección de la presentación que pide el cliente al servidor
- Protocolos trasporte: Protocolos por los que va a ir encapsulado el flujo multimedia

Por otro lado, se genera otro fichero relativo a las medidas de calidad, calculadas sobre los flujos de vídeo. Hemos configurado nuestro programa de manera que puede emitir estadísticas de dos maneras diferentes. La primera opción es configurarlo para que emita estadísticas globales al terminar la sesión. De esta manera, contaremos con unas medidas de la pérdida de paquetes durante toda la trasmisión del vídeo. La segunda opción es que el programa saque estadísticas de manera periódica cada 1 minuto, de este modo tendremos un reporte de la calidad de la trasmisión en tiempo real.

El fichero llamado stats.txt contendrá el siguiente formato:

- ID Sesión: Número de identificación único de cada sesión RTSP a la que pertenece el flujo multimedia

- Medidas sobre los paquetes que contienen en su *Transport Stream* algún TS perteneciente a nuestro flujo de vídeo:
    - Numero de paquetes capturados
    - Numero de paquetes perdidos (Estimación)

    - Porcentaje de pérdida de paquetes
- Medidas sobre los TS del flujo de vídeo detectado:
    - Numero de paquetes TS capturados
    - Numero de paquetes perdidos TS (Estimación)
    - Porcentaje de pérdida de paquetes TS

4.6 Estructuras de datos del programa

Una vez se ha explicado detalladamente cómo funcionan todos los módulos de nuestro programa, vamos a explicar con más detalle cómo hemos diseñado las diferentes estructuras de datos de nuestro programa

Nuestro programa cuenta con dos tablas hash: la primera indexada con la dupla del paquete y la segunda con el identificador de sesión. Al utilizar una tabla hash es posible que dos entradas diferentes produzcan la misma salida dando lugar a las colisiones. Para evitar este problema se producen listas enlazadas a cada entrada de la tabla.
Cada tabla guarda un puntero a una estructura de datos con toda la información extraída de la sesión o del flujo multimedia. En la Figura 18 se muestra un esquema de esta estructura: a la izquierda aparece la tabla de sesiones con las respectivas listas enlazadas representadas mediante círculos y, a la derecha, la tabla que guardará los distintos flujos de video. Las flechas representan los punteros a las distintas estructuras.

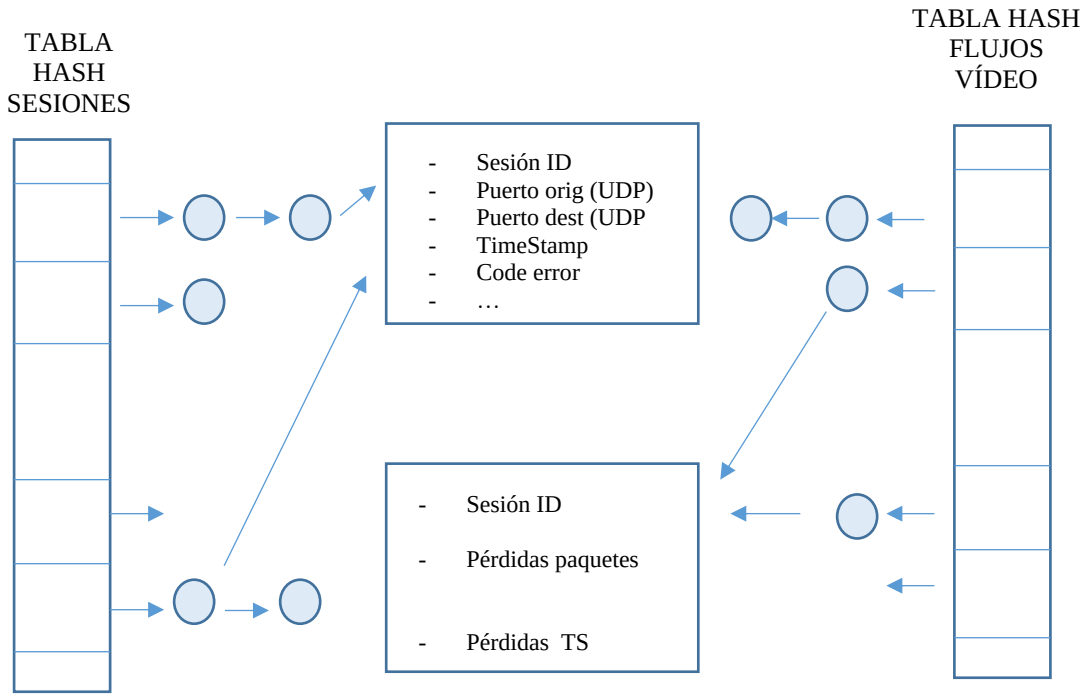

Figura 18. Estructura de datos para almacenar datos de la sesión multimedia.

Este esquema es el que nos va a permitir que nuestro programa puede monitorizar varios flujos y sesiones concurrentes. Es decir, que pueda mantener información de manera controlada y organizada de diferentes sesiones y flujos multimedia.

4.7 Conclusión

En este capítulo hemos explicado el desarrollo de nuestro programa, centrándonos sobre todo en la parte más técnica. Como hicimos en el capítulo de diseño, hemos divido nuestro programa en cuatro módulos principales y los hemos descrito detalladamente. Además, hemos hablado sobre como mostramos los datos que proporciona el programa, y sobre cómo se relacionan las diferentes estructuras de datos para tener una visión más global del funcionamiento de la herramienta. También hemos hablado En el siguiente capítulo, detallaremos la validación de nuestro código.

5. Integración, pruebas y resultados

5.1 Introducción

En este capítulo describiremos las pruebas que hemos realizado para comprobar que nuestro programa funciona correctamente y refleja el comportamiento real de la sesión. En primer lugar, explicaremos como hemos conseguido los ficheros de red que hemos utilizado como entrada a nuestro programa. La etapa de validación de nuestro código la dividiremos en dos partes: pruebas de análisis de la sesión RTSP, pruebas sobre las pérdidas en los flujos multimedia. En relación a las pruebas de análisis de la sesión hemos estudiado dos casos diferentes con dos códigos de estado distintos. Para las pérdidas en flujos multimedia hemos probado en dos casos con dos porcentajes de pérdidas diferentes.

5.2 Obtención de ficheros de validación

A la hora de obtener diferentes ficheros de red, para utilizar como entrada de nuestro programa, y que contuvieran sesiones RTSP, utilizamos dos métodos diferentes. Posteriormente, validamos nuestro código para lo que utilizamos el programa Wireshark para abrir nuestros ficheros .pcap

- VLC: es un reproductor multimedia libre que se puede configurar como servidor multimedia. Para recrear una sesión de *streaming* abrimos el VLC en un ordenador y lo configuramos como servidor (ver Figura 19). En otro ordenador abrimos el reproductor y lo configuramos como cliente. VLC nos permite utilizar señalización RTSP y lo único que tenemos que hacer es que el cliente a través de la IP del servidor le pida una solicitud de una presentación.

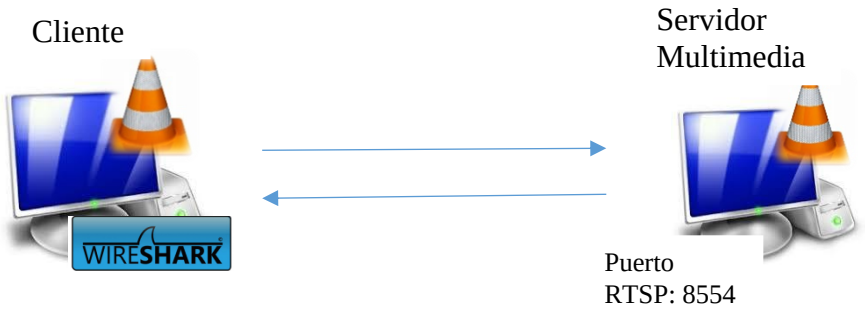

Figura 19. Esquema captura con VLC

- *Packet broker*: Este dispositivo lo hemos utilizado para capturar en el hogar el tráfico de red entre el decodificador de la televisión, que actuará como cliente, y el servidor multimedia. De esta manera, conseguiremos una captura del tráfico de red, y por tanto un modelo de señalización RTSP más realista. En las capturas realizadas el proveedor de red que proporciona el servicio es Movistar.
  En la Figura 20 se explica el esquema utilizado para capturar el tráfico en el hogar.

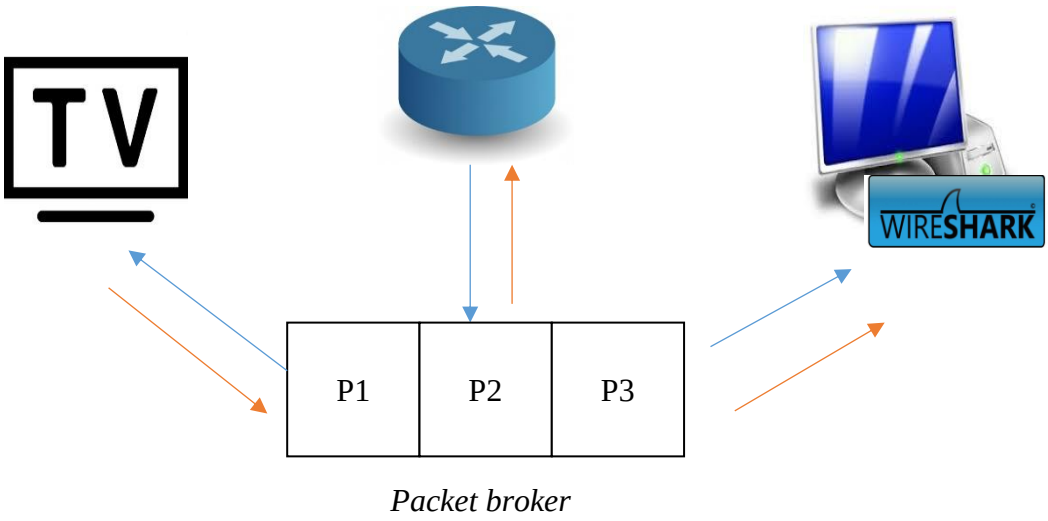

Figura 20. Esquema del sistema para capturar el tráfico de red en el hogar.

5.3 Pruebas de validación

En este apartado explicaremos las pruebas de realización que hemos realizado para verificar el correcto funcionamiento de nuestro código. A continuación, vamos a mostrar cual va a ser el formato de salida de nuestros ficheros con los datos extraídos. Todos los campos están separados por barras verticales para facilitar que el fichero pueda ser procesado por otro programa. En el caso de que un campo contenga varios valores, estos estarán separados por puntos y coma.

- Fichero resultante del análisis de una sesión RTSP:

```
| ID_sesión| IP SERVIDOR| IP CLIENTE| PUERTO CLIENTE| PUERTO
SERVIDOR| PROTOCOL TRANSPORT| USER AGENT|
SERVIDOR|METHOD_COMUNICATION| DURATION| METHODS_LIST| METHODS |
TIME| TIME_REPLY| CODE| URL| TRANSPORT_PROTOCOL_STREAM|
```

-Fichero resultante de las pérdidas en flujos multimedia:

```
| ID_sesión | Número_paquetes_TOTALES|
Número_de_paquetes_PERDIDOS| Porcentaje_paquetes_perdidos|
Número_de_TS| Número_de_TSLost| Porcentaje_TS_perdidos
```

A continuación, vamos a detallar el desarrollo de las pruebas de validación, para ello vamos a dividir esta sección en dos partes: análisis de sesión RTSP y pérdidas de paquetes en un flujo multimedia.

5.3.1 Análisis de sesión

1. Ejemplo de sesión básica sin errores :

El primer caso a estudiar es el de una sesión normal, en el cual el cliente envía una petición sobre una presentación, y tras ponerse de acuerdo en la configuración de los parámetros de trasporte del flujo multimedia, se comienza a trasmitir. En esta sesión se puede comprobar que no ha habido ningún tipo de error en la señalización puesto que el servidor contesta siempre a la petición del cliente con un 200 OK. En la Figura 21 se puede ver nuestra traza de red abierta desde Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
162816	472.956465	192.168.1.200	172.26.83.13	RTSP	122	OPTIONS * RTSP/1.0
162818	472.959397	172.26.83.13	192.168.1.200	RTSP	173	Reply: RTSP/1.0 200 OK
162825	472.964368	192.168.1.200	172.26.83.13	RTSP	348	DESCRIBE rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/2015
162828	472.970768	172.26.83.13	192.168.1.200	RTSP/S…	446	Reply: RTSP/1.0 200 OK
162829	472.975801	192.168.1.200	172.26.83.13	RTSP	407	SETUP rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/2015-10
162834	472.997025	172.26.83.13	192.168.1.200	RTSP	275	Reply: RTSP/1.0 200 OK
162835	472.999001	192.168.1.200	172.26.83.13	RTSP	431	PLAY rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/2015-10-(
162839	473.015870	172.26.83.13	192.168.1.200	RTSP	153	Reply: RTSP/1.0 200 OK
162960	473.333848	192.168.1.200	172.26.83.13	RTSP	383	PAUSE rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/2015-10
162961	473.341601	172.26.83.13	192.168.1.200	RTSP	152	Reply: RTSP/1.0 200 OK
163003	473.661990	192.168.1.200	172.26.83.13	RTSP	426	PLAY rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/2015-10-(
163004	473.670296	172.26.83.13	192.168.1.200	RTSP	153	Reply: RTSP/1.0 200 OK
173704	503.007924	192.168.1.200	172.26.83.13	RTSP	439	GET_PARAMETER rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/
173713	503.016827	172.26.83.13	192.168.1.200	RTSP	163	Reply: RTSP/1.0 200 OK (text/parameters)
185941	533.010662	192.168.1.200	172.26.83.13	RTSP	439	GET_PARAMETER rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/
185943	533.019025	172.26.83.13	192.168.1.200	RTSP	163	Reply: RTSP/1.0 200 OK (text/parameters)
188132	539.218640	192.168.1.200	172.26.83.13	RTSP	393	TEARDOWN rtsp://cdvr1.catchup.imagenio.telefonica.net:554/cutv/2015

Figura 21. Captura wireshark de sesión sin errores

A continuación, mostramos el fichero que genera nuestro programa al terminar la sesión. El puerto del servidor por el que se va a trasmitir el vídeo no se indica en la respuesta del servidor a la petición SETUP, y por ello, se escribe que no ha sido encontrado. También se puede observar que como explicamos en el capítulo 4.2 la respuesta al TEARDOWN nunca llega a producirse, es por ello que mientras que hay nueve métodos diferentes tenemos ocho tiempos de respuesta y códigos diferentes. Este problema lo solventaremos analizando el cierre de sesión a nivel de TCP.

| 3100546342062815915| 192.168.1.200| 172.26.83.13| 27249| NOT_FOUND| TCP| MICA-IP-STB| | unicast| 66.264786| DESCRIBE, SETUP, PLAY, PAUSE, GET_PARAMETER, TEARDOWN|.OPTIONS,DESCRIBE,SETUP,PLAY,PAUSE,PLAY,GET_PARAMETER,GET_PARAMETER,TEARDOWN|804401.847950;1443804401.855853;1443804401.867286;1443804401.890486;1443804402.225333;1443804402.553475;1443804431.899409;1443804461.902147;1443804468.110125|804401.850882;1443804401.862253;1443804401.888510;1443804401.907355;1443804402.233086;1443804402.561781;1443804431.908312;1443804461.910510| 200 OK;200 OK;200 OK;200 OK;200 OK;200 OK;200 OK;200 OK| rtsp://172.26.83.13:5554/cutv/2015-10-02T17:16:41Z/891c6e2409a64ffc458a449102923fbca0fd588dfdc3a009535868dc57ae6d48/rolling_buffer/1705/2015-10-02T15:54:00Z/2015-10-02T17:01:00Z/3100546342062815915| MP2T/H2221/UDP|

2. Ejemplo de sesión con el código 404 Client error

En este tercer escenario, el cliente pide una presentación a través de la petición DESCRIBE que el servidor multimedia no reconoce como una propia. En consecuencia, la respuesta a esta petición, en vez de contener los parámetros de inicialización de la sesión, responde con un código de error. Para recrear esta situación, nos hemos servido del reproductor VLC y lo hemos configurado en modo cliente/servidor. Al ordenador que hacía de cliente, le hemos pedido que solicite una presentación que no se encontraba en el servidor.

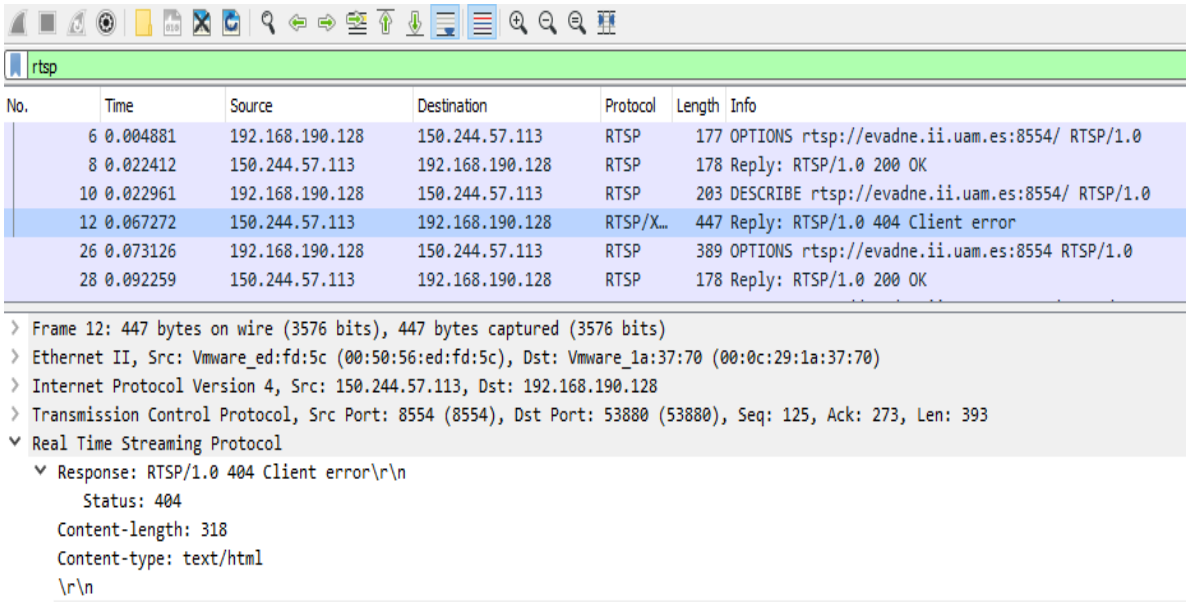

Figura 22 Captura Wireshark de sesión con código 404 Client error

A continuación, mostramos el fichero que genera nuestro programa. Esta sesión termina una vez recibida la respuesta con el código 404 Client error. Los espacios en blanco se debe a que esos datos se extraían de información posterior a la petición DESCRIBE.

| | | 192.168.190.128| 150.244.57.113| | | | LibVLC/2.1.6 (LIVE555 Streaming Media v2014.01.13)| | | | OPTIONS,DESCRIBE,| 887401.486606;1526887401.504686| 887401.504137;1526887401.548997| 200 OK;404 Client error| | |

5.3.2 Pérdidas en flujos multimedia

En esta parte explicaremos las pruebas que hemos realizado para comprobar que nuestra modelo de calidad de los flujos multimedia funcionaba correctamente.

Lo primero que hicimos fue producir pérdidas controladas en nuestras trazas de red, para posteriormente corroborar que nuestra herramienta las estimaba adecuadamente. Para introducir estas pérdidas diseñamos un programa que por cada paquete llama a una función aleatoria (random) que genera un número del 0 al 1. Si el valor obtenido es menor que el porcentaje de pérdidas en tanto por uno introducido, el paquete se desechará. Posteriormente, a través del comando editcap, eliminaremos los paquetes calculados en el programa anterior. Este sistema nos proporcionaría unas pérdidas en las capturas iguales al porcentaje introducido si el número de paquetes introducido fuera infinito. En un caso real, con un número de paquetes finitos, obtendremos un porcentaje muy similar al pedido pero no idéntico.

1. Introducimos unas pérdidas 0.5 %

Para este caso las pérdidas introducidas por nuestro programa en la traza de internet con respecto a todos los paquetes son del 0.524 %.
A continuación, mostramos una imagen del vídeo con pérdidas. Al ser unas pérdidas muy pequeñas se aprecian menos en la imagen. Con flechas naranjas hemos indica los pixeles en los que se aprecian las pérdidas en la captura.

Figura 23. Imagen con 0.05% de pérdidas

Para poder apreciar mejor la diferencia mostramos una imagen del vídeo antes de introducir las pérdidas.

Figura 24. Imagen sin pérdidas

En las siguientes líneas se muestra el fichero generado con las estadísticas. En él está subrayado el porcentaje de pérdidas que hemos detectado. El porcentaje de pérdidas en paquetes de flujo de vídeo es de 0.005168 en tanto por uno. Hay que tener en cuenta, que las pérdidas introducidas a nuestra captura de tráfico de red, se aplican a todos los paquetes y, por tanto, no se podrá saber con exactitud el porcentaje exacto de paquetes de flujo de vídeo perdidos.

|3100545326303049988|95674.000000|497.000000|0.005168|490523.000000|
2479.000000|0.005028

2. Introducimos unas pérdidas 5 %

Para este caso las pérdidas introducidas por nuestro programa en la traza de internet con respecto a todos los paquetes son del 5,002 %.
A continuación, mostramos una imagen del vídeo con pérdidas. En este caso, las pérdidas son mucho mayores con respecto al caso anterior y de ahí que la calidad de la imagen no sea buena.

Figura 25 Imagen con 0.5% de pérdidas

Ahora se muestra una imagen del vídeo sin pérdidas para apreciar la diferencia

Figura 26.Imagen sin pérdidas

Ejemplo salida archivo:

|310054532630304998 8|91424.000000|4705.000000|0.048945|464675.000000|239 51.000000|0.049017

En el anexo B mostramos un ejemplo del archivo de salida cuando se configura el programa para reportar las pérdidas de manera periódica en vez de al final de la sesión. En este ejemplo se programó la herramienta de manera que se realizan las medidas de calidad cada 10 segundos.

5.4 Conclusión

En este capítulo, hemos explicado cómo se han obtenido los archivos de validación necesarios para poder comprobar que nuestro programa funciona de manera eficiente antes distintos escenarios. Además, se ha introducido una forma nueva de generar un tanto por ciento de pérdidas en una captura de red.
De las distintas pruebas realizadas, podemos concluir que nuestro programa proporciona un reporte correcto de los sucesos más importantes ocurridos durante la sesión RTSP. Además, es capaz de identificar por donde se está trasmitiendo el flujo de vídeo y realizar las medidas de calidad oportunas. En el análisis de estas medidas de calidad, se ha podido comprobar que el valor de pérdidas estimado se corresponde con las pérdidas introducidas a nuestra captura. En el siguiente capítulo procederemos a resumir las conclusiones globales del trabajo y los posibles trabajos futuros que se pueden realizar.

6. Conclusiones y trabajo futuro

6.1 Conclusiones

El objetivo de nuestro TFG ha sido la realización de una herramienta de monitorización de los servicios de vídeo sobre redes IP a través del análisis de la monitorización RTSP. De esta manera, se ha desarrollado un programa que detecta cuando un cliente solicita un contenido multimedia al servidor. A partir de este instante, se monitoriza la conexión que se establece con el servidor multimedia y se detecta si ha habido algún tipo de incidencia durante la comunicación. Además, nuestro programa identifica cuándo y por donde se empieza a trasmitir los flujos multimedia y calcula si se ha producido alguna pérdida durante la trasmisión. Por último, se emitirán unos archivos con los datos y las estadísticas extraídas de cada sesión

Nuestro trabajo lo hemos diseñado separándolo en cuatro módulos diferentes, que nos han servido para estructurar nuestro proyecto. Además, hemos basado nuestro diseño en los requisitos tanto funcionales como no funcionales que necesitamos cumplir.

Una vez desarrollado nuestro programa, hemos validado su funcionamiento. En primer lugar, hemos intentado recrear varios escenarios de una conexión entre el cliente y el servidor multimedia. Hemos recreado un escenario en el que el cliente solicita una presentación que el cliente no tenía y hemos corroborado que nuestro programa detecta la incidencia. Por otro lado, hemos visto que, ante una conexión básica, nuestro programa es capaz de monitorizar todas las peticiones y los instantes en lo que se envían, además de detectar por dónde se trasmite el flujo multimedia. Por otro lado, hemos validado que el modelo desarrollado de contabilidad de las pérdidas del flujo multimedia basado en el *Continuity Counter* (campo de la cabecera MPEG-TS) estima correctamente. Por tanto, podemos concluir que en el caso de que nuestra herramienta se utilizara para monitorizar el tráfico de video en un hogar, podríamos saber en tiempo real la calidad del servicio que se está prestando. Hay que tener en cuenta, que nuestro sistema sirve para la visualización de vídeos bajo demanda del usuario, en los cuales el cliente tiene el control de la sesión (pausar, reproducir, rebobinar, etc.).

Por último, concluir que en este trabajo se han podido afianzar diferentes conceptos que se ya se habían introducido en otras asignaturas de la carrera. A nivel de programación han sido muy útiles, por supuesto, los conocimientos aprendidos en materias como Programación 1 y 2 y Programación Orientada a Objetos, en las que se aprendió a programar y a utilizar diferentes algoritmos. Por otro lado, las asignaturas de Redes, en concreto Redes Multimedia de tercero, nos han ayudado a entender bien los diferentes protocolos de trasporte y de aplicación vistos durante el desarrollo de este trabajo. Por último, es conveniente hacer hincapié en la asignatura de Sistemas y Servicios Audiovisuales, que nos ha permitido tener las bases para poder realizar las diferentes medidas sobre los flujos multimedia.

6.2 Trabajo futuro

En vistas a poder ampliar el trabajo realizado en este trabajo proponemos tres posibles continuaciones que desde nuestro punto de vista serían interesantes de realizar:

- Desarrollar un módulo que a partir de los datos de pérdidas obtenidos por nuestro programa estime la calidad de la experiencia en la escala MOS siguiendo un método similar a los utilizados en la referencia [8].

- Continuar con el análisis de los flujos de vídeo, proporcionando un análisis del códec H.264. De esta manera, se podría ajustar las medidas de calidad a las características de este modelo. Como resultado se podrán definir si las pérdidas halladas son relativas a un cuadro de tipo I, P o B, pues su pérdida tiene una incidencia desigual sobre el MOS [9].

- Realizar una monitorización de los servicios de vídeo sobre redes IP, pero en este caso para trasmisiones *multicast*. Para ello será necesario analizar el protocolo IGMP que nos dará información sobre este tipo de comunicaciones.

- Probar el rendimiento de la herramienta en una red de alta velocidad simulando el escenario de la red troncal de un operador.

7. Referencias

[1] M. r Fernando Boronat Seguí, IPTV, la televisión por internet.

[2] A. R. H. Schulzrinne, «RTSP, RFC2326,» 1998. [En línea]. Available: https://tools.ietf.org/html/rfc2326.

[3] «Byteway,» Multiplexing of MPEG-2 streams, [En línea]. Available: https://byteway.wordpress.com/2014/01/18/multiplexing-of-mpeg-2-streams/.

[4] «MPEG-2 para DVB [MPEG-2 "Flujo de Transporte"],» [En línea]. Available: http://62.97.113.75/wiki/MPEG_2.

[5] «MPEG2-Transmission,» [En línea]. Available: http://www.erg.abdn.ac.uk/future-net/digital-video/mpeg2-trans.html.

[6] H. Schulzrinne, «RTP RFC 3550,» Julio 2003. [En línea]. Available: https://tools.ietf.org/html/rfc3550.

[7] «TCP RFC 793,» 1981. [En línea]. Available: rfc793.

[8] D. Hernando Loeda, «Desarollo de un sistema de medición, monitorización y gestión de redes IPTV,» 2013.

[9] A. Mozo Robles, «Diseño e implementación de un sistema de medición y monitorización de la calidad de servicio y experiencia de redes IPTV basadas en el codec H.264,» TFM.

[10] J. V. d. Meer, Fundamentals and Evolution of MPEG-2 Systems: Paving the MPEG Road, John Wiley & Sons, June 3, 2014.

[11] I. E. Richardson, The H.264 Advanced Video Compression Standard, Second Edition, John Wiley & Sons, August 17, 2010.

[12] J. H. Baxter, Y. Orzach y C. Mishra, «Troubleshooting RTSP,» de *Wireshark Revealed: Essential Skills for IT Professionals*, December 15, 2017.

[13] W. Simpson y H. Greenfield, IPTV and Internet Video, 2nd Edition, Focal Press, November 12, 2012.

[14] Digital Video and Audio Broadcasting Technology: A Practical Engineering Guide.

[15] «wirexhark,» [En línea]. Available: https://www.wireshark.org/docs/man-pages/editcap.html.

[16] «cubro,» [En línea]. Available: https://www.cubro.com/packetmaster-ex2.html.

[17] D. Mateos Costilla y S. Reaño Montoro, «Streaming de Audio/Video. Protocolo RTSP,» [En línea]. Available: http://edicions.uib.cat/ojs/index.php/enginy/article/viewFile/74/53.

[18] A. Martín Hernandez, «Desarrollo de un sistema de medida de recogida de Datos y monitorización de trafico VOIP,» TFG.

[19] H. Schulzrinne, 2003. [En línea]. Available: https://tools.ietf.org/html/rfc3550.

[20] «UDP RFC 768,» 1980. [En línea]. Available: https://tools.ietf.org/html/rfc768.

[21] [En línea]. Available: https://www.enmimaquinafunciona.com/pregunta/53877/streaming-de-medios-de-comunicacion-desde-el-interior-de-las-paginas-html-por-ejemplo.

8. Anexos

A. *Script* de generación de pérdidas en archivos pcap

```
2
3   import random
4   import sys
5
6   LOSS = float(sys.argv[1])
7   #LOSS = 0.05
8   contador=0
9
10  f = open ("fichero.txt", "w")
11
12  for  x in range(0, 10000):
13          if random.random() < LOSS:
14              f.write(str(x)+" "),
15              contador+=1
16
17  f.write("_____________Fin_____________--")
18  print("NUMERO:",contador)
19
20
21
```

El script anterior genera un fichero con el número de paquetes que decidimos eliminar. A continuación utilizaremos el comando editcap para quitar los paquetes obtenidos de nuestra traza de red. El formato a utilizar será el siguiente:

editcap capture.pcap exclude.pcap nºpaquetes

En nuestro caso hemos utilizado el siguiente comando seguido de la lista de paquetes obtenida por nuestro programa.

- editcap movistar_sesion_p0.pcap movistar_sesion_p1.pcap 317 589 710 781 786 ...

B. Ejemplo de salida de estadísticas de calidad cada 10 segundos

```
-------- Estadísticas para sesion 3100545326303049988 ----------
-
------- Frame 317 -----------
Número de paquetes TOTALES: 1967.000000
Número de paquetes PERDIDOS (Estimación): 100.000000
Porcentaje paquetes perdidos: 0.048379
Número de TS: 10028.000000
Número de TSLost: 518.000000
Porcentaje TS perdidos: 0.049118
-------- Estadísticas para sesion 3100545326303049988 ----------
-
------- Frame 722 -----------
Número de paquetes TOTALES: 2057.000000
Número de paquetes PERDIDOS (Estimación): 96.000000
Porcentaje paquetes perdidos: 0.044589
Número de TS: 10425.000000
Número de TSLost: 487.000000
Porcentaje TS perdidos: 0.044630
-------- Estadísticas para sesion 3100545326303049988 ----------
-
------- Frame 985 -----------
Número de paquetes TOTALES: 2037.000000
Número de paquetes PERDIDOS (Estimación): 105.000000
Porcentaje paquetes perdidos: 0.049020
Número de TS: 10387.000000
Número de TSLost: 535.000000
Porcentaje TS perdidos: 0.048984
-------- Estadísticas para sesion 3100545326303049988 ----------
-
------- Frame 1396 -----------
Número de paquetes TOTALES: 2038.000000
Número de paquetes PERDIDOS (Estimación): 108.000000
Porcentaje paquetes perdidos: 0.050326
Número de TS: 10364.000000
Número de TSLost: 551.000000
Porcentaje TS perdidos: 0.050481
-------- Estadísticas para sesion 3100545326303049988 ----------
-
------- Frame 1858 -----------
Número de paquetes TOTALES: 2045.000000
Número de paquetes PERDIDOS (Estimación): 104.000000
Porcentaje paquetes perdidos: 0.048395
Número de TS: 10373.000000
Número de TSLost: 535.000000
Porcentaje TS perdidos: 0.049047
```